OPTIMASI KOMPOSISI PAKAN TERNAK SAPI MENGGUNAKAN ALGORITME GENETIKA - *SIMULATED ANNEALING*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:

Akhmad Syururi

NIM: 115060801111092

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

OPTIMASI KOMPOSISI PAKAN TERNAK SAPI MENGGUNAKAN ALGORITME GENETIKA - *SIMULATED ANNEALING*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Akhmad Syururi
NIM: 11506080111092

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Nurul Hidayat, S.Pd, M.Sc	Ratih Kartika Dewi, S.T., M.Kom
NIP: 19680430 200212 1 001	NIK: 201503 890520 2 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 3 Agustus 2018

Akhmad Syururi
NIM: 115060807111092

KATA PENGANTAR

Segala puji bagi Allah Ta'ala, Rabb seluruh alam yang telah memberikan hidayah taufiq dan kemudahan sehingga skripsi yang berjudul **"Optimasi komposisi pakan ternak sapi menggunakan metode Algoritme Genetika dan Simulated Aneling"** dapat terselesaikan. Shalawat serta salam semoga tercurah kepada Nabi Muhammad SAW., keluarganya, para sahabatnya, dan orang-orang yang mengikuti mereka dengan baik hingga hari kiamat.

Penyusunan skripsi ini dapat terselesaikan dengan bantuan banyak pihak, oleh karena itu pada kesempatan ini penulis ingin menyampaikan ucapan terimakasih kepada:

1. Nurul Hidayat, S.Pd, M.Sc dan Ratih Kartika Dewi, S.T., M.Kom selaku dosen pembimbing skripsi yang telah membimbing dan mengarahkan penulis sehingga skripsi ini dapat terselesaikan.
2. Agus Wahyu Widodo, S.T, M.Cs, selaku ketua program studi informatika yang telah memberikan kesempatan kepada penulis untuk menyelesaikan skripsi ini.
3. Kedua orangtua penulis, Alm. H.M.Zainal Arifin, Hj Mas Udah, dan Kakak saya yang selalu mendukung penulis baik berupa doa, motivasi, kasih sayang, nasihat, moril dan materil yang telah diberikan kepada penulis.
4. Teman seangkatan saya yang selalu sabar dalam mendukung penulis, memberi semangat, dan doa.
5. Isna Besti Sofia calon pendamping saya yang selalu mendukung dari doa ,suport dan membantu saya dalam penyelesaian penulisan skripsi ini.
6. Seluruh dosen informatika atas kesabaran dan dedikasinya dalam mengajarkan ilmunya kepada penulis.
7. Eko Abdul Prakoso yang selalu memberi motivasi dan memfasilitasi penulis dalam menyelesaikan skripsi ini.
8. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya skripsi ini.

Malang, 19 juli 2018

Penulis
Akhmad.syururi92@gmail.com

ABSTRAK

Konsumsi daging sapi di Indonesia terus mengalami peningkatan, namun peningkatan tersebut belum diimbangi dengan penambahan produksi yang memadai. Laju peningkatan populasi sapi potong relatif lamban, yaitu 4,23% pada tahun 2007. Produktivitas ternak dipengaruhi oleh faktor pakan, manajemen dan pembibitan. Permasalahan yang sering dihadapi peternak adalah ketersediaan pakan, yang mana pakan yang diberikan pada ternak memiliki kandungan nutrisi yang rendah sehingga akan memengaruhi kecernaan pakan dan ketersediaan nutrisi yang berdampak pada rendahnya produktivitas ternak. Berdasarkan kasus yang terjadi, maka sistem yang dibangun pada penelitian ini adalah "optimasi komposisi pakan ternak sapi menggunakan algoritme genetika - *simulated annealing*". Metode yang digunakan dalam optimasi komposisi pakan ternak sapi pada sistem ini menggunakan algoritme genetika - *simulated annealing*. Tujuan penggabungan algoritme genetika - *simulated annealing* (SA) adalah untuk memanfaatkan kelebihan *simulated annealing* yang mampu bertahan menghadapi lokal optimum dan proses pencarian yang dikendalikan oleh suhu untuk digunakan dalam menutupi kekurangan algoritme genetika tersebut. Hasil Dari pengujian yang telah dilakukan ternyata semakin kecil nilai *Popsize* dan iterasi, semakin besar nilai *tn* dan *t0* maka semakin besar kemungkinan algoritme genetika - *simulated annealing* untuk mendapatkan hasil yang lebih baik daripada algoritme genetika saja, sebaliknya semakin besar nilai *Popsize* dan iterasi, semakin kecil nilai *tn* dan *t0* maka semakin kecil pula kemungkinan algoritme genetika - *simulated annealing* untuk mendapatkan hasil yang lebih baik.

Kata kunci: *optimasi, algoritme genetika, simulated annealing*

ABSTRACT

Beef consumption in Indonesia keeps increasing, but the increase has not been balanced with the addition of an adequate production. The rate of population increase beef cattle is relatively slow, 4.23% in 2007. The productivity of the livestock affected by the feed factors, management and breeding. The problem often encountered breeders is the availability of feed, which feed given to livestock contain nutrients that are low so will affect the availability of feed and digestibility of nutrients that impact low produkt ivitas cattle. Based on cases occurs, then the system is built on this research is "beef cattle feed composition optimization using genetic algorithm and simulated annealing". Methods used in beef cattle feed composition optimization on this system using a genetic algorithm and simulated annealing. The purpose of merging algorithm and genetic simulated annealing (SA) is to make use of the advantages of simulated annealing that are able to survive are facing local optimum and the search process which is controlled by the temperature to be used in cover lack of genetic algorithms. The results of the testing that has been done progressively smaller values Popsize and iterations, the greater the value of tn and t0 then the more likely genetic algorithm and simulated annealing to get better results than the algorithm Genetics alone, otherwise the larger value of Popsize and iterations, the smaller value of tn and t0 the small possibility of genetic algorithm and simulated annealing for getting better results., genetic algorithm, simulated annealing

Kata kunci: *Optimization ,Genetic Algorithm, simulated annealing*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BAB I PENDAHULUAN

1.1 Latar Belakang

Konsumsi daging sapi di Indonesia terus mengalami peningkatan. Namun peningkatan tersebut belum diimbangi dengan penambahan produksi yang memadai. Laju peningkatan populasi sapi potong relatif lamban, yaitu 4,23% pada tahun 2007 (Direktorat Jendral Peternakan 2007). Kondisi tersebut menyebabkan sumbangan sapi potong terhadapproduksi daging nasional rendah (Mersyah 2005; Santi 2008) sehingga terjadi kesenjangan yang makin lebar antara permintaan dan penawaran (Setiyono et al. 2007). Pada tahun 2006, tingkat konsumsi daging sapi diperkirakan 399.660 ton, atau setara dengan 1,70−2 juta ekor sapi potong (Koran Tempo 2008), sementara produksi hanya 288.430 ton. Pemerintah memproyeksikan tingkat konsumsi daging pada tahun 2010 sebesar 2,72 kg/kapita/tahun sehingga kebutuhan daging dalam negeri mencapai 654.400 ton dan rata-rata tingkat pertumbuhan konsumsi 1,49%/tahun (Badan Pusat Statistik 2005)

Produktivitas ternak dipengaruhi oleh faktor pakan, manajemen dan pembibitan. Permasalahan yang sering dihadapi peternak adalah ketersediaan pakan, dimana pakan yang diberikan pada ternak memiliki kandungan nutrisi yang rendah. Sebagai contoh adalah limbah hasil samping pertanian, industri pertanian dan pangan. Limbah pertanian berasal dari limbah tanaman pangan seperti jerami jagung, jerami padi dan lain-lain. Kandungan nutrisi yang rendah akan mempengaruhi kecernaan pakan dan ketersediaan nutrisi sehingga produktivitas ternak juga rendah (Wahyono, 2011).

Sumber utama daging sapi nasional masih tergantung pada usaha pembibitan di dalam negeri yang berupa peternakan rakyat. Sampai saat ini belum ada perusahaan swasta atau perusahaan negara yang bergerak di bidang pembibitan sapi karena dinilai usaha tersebut kurang menguntungkan. Sehingga peternakan sapi potong rakyat merupakan tulang punggung bagi perkembangan peternakan sapi di Indonesia (Hadi dan Ilham, 2002).

Gejala kekurangan pakan ini kemungkinan terjadi setelah lepas sapih, dimana sapi sudah tidak mendapatkan air susu dan konsumsi pakannya sangat tergantung kepada pakan yang disediakan oleh peternak (Santosa, 1985).

Besarnya prosentase penggunaan pakan yang tidak efisien menunjukkan bahwa pakan ternak (ransum) menempati posisi penting dalam usaha peternakan. Dalam sudut pandang ekonomi, biaya untuk pembelian pakan ternak merupakan biaya tertinggi dalam usaha peternakan, sehingga biaya tersebut harus ditekan serendah mungkin untuk memaksimalkan pendapatan (Nugraha, 2011).

Permasalahan penyusunan komposisi bahan pakan dapat diselesaikan dengan metode optimasi. Dalam penelitian ini, permasalahan optimasi bahan pakan sapi potong dapat diselesaikan dengan algoritme genetika, hal itu dikarenakan algoritme genetika memiliki kelebihan dalam menghasilkan output yang optimal dengan tetap memperhatikan faktor nutrisi dan biaya. Penggunaan konsep evolusi biologi akan menghasilkan suatu output berupa komposisi bahan pakan yang sebaiknya dikonsumsi untuk memenuhi kebutuhan nutrisi (Aribowo, 2008).

Salah satu penelitian yang menggunakan Algortitma Genetika dan *Simulated annealing* yaitu penelitian yang dilakukan oleh Fitri Anggarsari dengan judul Optimasi Kebutuhan Gizi untuk Balita Menggunakan Hybrid Algoritme genetika - *simulated annealing*, dari penelitian tersebut dihasilkan kesimpulan bahwa Algoritme genetika - *simulated annealing* dapat digunakan untuk menentukan komposisi makanan yang sesuai untuk kebutuhan gizi balita.

Berdasarkan pemaparan penelitian-penelitian sebelumnya maka penulis akan melakukan penelitian tentang Algoritme genetika - *simulated annealing* untuk optimasi komposisi pakan ternak sapi. Sistem menyediakan masukan berupa data tentang jenis sapi, kebutuhan gizi, dan daftar makanan yang kemudian akan diproses dengan Algoritme genetika - *simulated annealing* hingga menghasilkan keluaran berupa komposisi makanan dengan gizi terbaik dan harga terendah.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, maka diperoleh rumusan masalah yang mendasari penelitian ini adalah sebagai berikut :

1. Bagaimana implementasi algoritme genetika - *simulated annealing* untuk optimasi komposisi pakan ternak sapi?
2. Bagaimana hasil pengujian algoritme genetika - *simulated annealing* untuk optimasi komposisi pakan ternak sapi?

1.3 Tujuan

Tujuan pada penelitian ini adalah sebagai berikut :

1. Mengimplementasikan algoritme genetika - *simulated annealing* untuk optimasi komposisi pakan ternak sapi.
2. Menguji algoritme genetika - *simulated annealing* untuk optimasi komposisi pakan ternak sapi.

1.4 Manfaat

Manfaat yang diharapkan dari penelitian ini adalah dapat memberikan informasi komposisi pakan ternak sapi sehingga peternak dapat mengkombinasikan pakan dengan komposisi yang baik untuk sapi.

1.5 Batasan Masalah

Agar permasalahan yang dirumuskan dapat lebih berfokus dan tidak meluas, maka batasan-batasan yang ditentukan pada penelitian ini adalah sebagai berikut:

1. Keluaran dari sistem ini berupa kombinasi pakan ternak sapi beserta nilai *Fitness*.
2. Menggunakan bahasa pemrograman JAVA.
3. Pengujian yang dilakukan adalah pengujian pengaruh setiap variabel terhadap hasil.

1.6 Sistematikan Pembahasan

Adapun sistematika penulisan yang digunakan dalam menyusun laporan ini adalah sebagai berikut :

BAB I PENDAHLUAN

Pada BAB ini memuat latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, sistematika penulisan skripsi yang terkait dengan optimasi pakan ternak sapi menggunakan algoritme genetika - *simulated annealing*.

BAB II TINJAUAN PUSTAKA

Membahas tentang kajian pustaka yang berhubungan dengan penelitian ini dan teori dasar tentang sapi, pakan ternak sapi, algoritme genetika - *simulated annealing*.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan metodologi dan proses analisa kebutuhan dan perancangan sistem yang digunakan dalam penelitian ini. Metode yang digunakan antara lain study literatur, pengumpulan data, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis sistem dan pengambilan kesimpulan.

BAB IV PERANCANGAN SISTEM

Membahas tentang kebutuhan dan perancangan pemodelan optimasi pakan ternak sapi menggunakan algoritme genetika - *simulated annealing*.

BAB V IMPLEMENTASI

Membahas tentang implementasi sistem sesuai dengan perancangan yang telah dibuat sebelumnya.

BAB VI PENGUJIAN DAN ANALISIS

Membahas tentang proses pengujian dan hasil dari optimasi pakan ternak sapi menggunakan algoritme genetika - *simulated annealing* dan juga anaisis dari pengujian yang dilakukan.

BAB VII PENUTUP

Membahas tentang kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak dalam pemodelan optimasi pakan ternak sapi menggunakan algoritme genetika - *simulated annealing* dan saran pengembangan penelitian selanjutnya.

BAB II TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Kajian pustaka pada penelitian ini akan membahas beberapa penelitian sistem pakar yang telah dilakukan sebelumnya. Beberapa penelitian ini akan digunakan penulis untuk mendukung penelitian dalam skripsi ini. Penelitian - penelitian tersebut ditunjukkan pada tabel 2.1.

Penelitian pertama dilakukan oleh Mahbub Zaeni Efendi yang berjudul "Sistem Penjadwalan Kuliah Ittelkom Dengan Hybrid Algoritme Genetika *Simulated annealing* (GA-SA)". Penelitian ini bertujuan untuk memudahkan perguruan tinggi dalam menyusun penjadwalan perkuliahan. Variabel yang dibutuhkan dalam penelitian ini adalah jumlah kelas dan ruang. Hasil dari penelitian ini berupa perangkat lunak yang dapat menyusun penjadwalan kuliah pada suatu perguruan tinggi dan pada penelitian ini menunjukkan bahwa terdapat kenaikan nilai *Fitness* pada proses GASA yang cukup signifikan untuk tiap generasinya.

Penelitian kedua dilakukan oleh Amalia Kartika Aryani yang berjudul "Hibridisasi Algoritme genetika - *simulated annealing* untuk Optimasi Multi-Trip Vehicle Routing Problem with Time Windows (Studi Kasus: Pariwisata Kabupaten Banyuwangi)". Penelitian ini bertujuan untuk mengatasi permasalahan Multi-trip VRPTW. Variabel yang dibutuhkan dalam penelitian ini adalah jam buka dan jam tutup untuk setiap daerah wisata. Hasil dari penelitian ini adalah sebuah rekomendasi kombinasi tempat yang dapat dikunjungi dalam 3 hari dan pada penelitian ini menunjukkan bahwa Hibridisasi algoritme genetika dan simulated annealing mampu menyelesaikan permasalahan Multi-trip VRPTW lebih optimal dari pada penggunaan algoritme genetika dan *simulated annealing* secara terpisah.

Penelitian ketiga dilakukan oleh Fitri Anggarsari yang berjudul "Optimasi Kebutuhan Gizi untuk Balita Menggunakan Hybrid Algoritme genetika - *simulated annealing*". Penelitian ini bertujuan untuk mengatasi permasalahan pemberian makanan sesuai gizi balita. Variabel yang dibutuhkan pada penelitian ini berupa macam-macam bahan makanan beserta zat yang terkandung di dalamnya. Hasil dari penelitian ini berupa perangkat lunak yang dapat memberikan rekomendasi bahan makanan sesuai kebutuhan gizi yang mendekati kebutuhan gizi balita yang sebenarnya dengan mempertimbangkan berat bahan makanan dan harga yang minimal dalam satu hari dan pada penelitian ini menunjukkan bahwa kebutuhan kalori balita yang sebenarnya belum sesuai atau belum mendekati dengan kebutuhan kalori yang direkomendasikan. Akan tetapi masih berada pada batas toleransi penetapan atau penyusunan kalori. Sehingga implementasi hybrid algoritme genetika dan *simulated annealing* ini mampu mengoptimasi kebutuhan gizi balita dengan baik.

Penelitian keempat dilakukan oleh Durrotul Fakhiroh yang berjudul "Optimasi Komposisi Pakan Sapi Perah Menggunakan Algoritme Genetika". Penelitian ini bertujuan untuk memberikan rekomendasi kombinasi pakan ternak dengan nutrisi terbaik dan harga terendah. Variabel yang dibutuhkan pada penelitian ini berupa macam-macam bahan pakan beserta nilai gizi dan harganya. Hasil dari penelitian ini berupa perangkat lunak yang dapat memberikan rekomendasi komposisi ransum dengan biaya yang minimal dan kebutuhan nutrisi sapi perah tetap terpenuhi dan pada penelitian ini menunjukkan bahwa representasi kromosom real code mampu menyelesaikan permasalahan optimasi komposisi pakan sapi perah. Algoritme genetika dalam permasalahan ini mampu menekan biaya dan memaksimalkan pemenuhan kebutuhan nutrisi.

Tabel 2.1 Tinjauan Pustaka

No	Judul	Objek (*Input*)	Metode (Proses)	Hasil (*Output*)
1	Sistem Penjadwalan Kuliah Ittelkom Dengan Hybrid Algoritme Genetika *Simulated annealing* (GA-SA)	jumlah kelas dan ruang	Algoritme genetika - *simulated annealing*	• penjadwalan kuliah pada suatu perguruan tinggi • terdapat kenaikan nilai *Fitness* pada proses GASA yang cukupsignifikan untuk tiap generasinya
2	Hibridisasi Algoritme genetika - *simulated annealing* untuk Optimasi Multi-Trip Vehicle Routing Problem with Time Windows (Studi Kasus: Pariwisata Kabupaten	jam buka dan jam tutup untuk setiap daerah wisata	Algoritme genetika - *simulated annealing*	• rekomendasi kombinasi tempat yang dapat dikunjungi dalam 3 hari • Hibridisasi algoritme genetika dan simulated annealing mampu menyelesaikan permasalahan Multi-trip

	Banyuwangi)			VRPTW lebih optimal dari pada GAZA

Tabel 2.2 Tinjauan Pustaka (lanjutan)

No	Judul	Objek (*Input*)	Metode (Proses)	Hasil (*Output*)
3	Optimasi Kebutuhan Gizi untuk Balita Menggunakan Hybrid Algoritme genetika - *simulated annealing*	macam-macam bahan makanan beserta zat yang terkandung di dalamnya	Algoritme genetika - *simulated annealing*	• rekomendasi bahan makanan sesuai kebutuhan gizi yang mendekati kebutuhan gizi balita • hybrid algoritme genetika dan simulated annealing ini mampu mengoptimasi kebutuhan gizi balita dengan baik
4	Optimasi Komposisi Pakan Sapi Perah Menggunakan Algoritme Genetika	macam-macam bahan pakan beserta nilai gizi dan harganya	Algoritme Genetika	• rekomendasi komposisi ransum dengan biaya yang minimal dan kebutuhan nutrisi sapi perah tetap terpenuhi • Algoritme genetika dalam permasalahan ini mampu menekan biaya dan memaksimalkan pemenuhan kebutuhan nutrisi

2.2 Sapi Potong

Sapi potong merupakan salah satu ternak penghasil daging di Indonesia. Namun, produksi daging sapi dalam negeri belum mampu memenuhi kebutuhan karena populasi dan tingkat produktivitas ternak rendah (Isbandi 2004; Rosida 2006; Direktorat Jenderal Peternakan 2007; Syadzali 2007; Nurfitri 2008; Santi 2008). Rendahnya populasi sapi potong antara lain disebabkan sebagian besar ternak dipelihara oleh peternak berskala kecil dengan lahan dan modal terbatas (Kariyasa 2005; Mersyah 2005; Suwandi 2005).

Berdasarkan data sebaran populasi sapi potong di Indonesia tahun 2007 (Direktorat Jenderal Peternakan 2007), sentra sapi potong terdapat di Jawa Timur, Jawa Tengah, Nanggroe Aceh Darussalam (NAD), Bali, Nusa Tenggara Timur, Sumatera Selatan, dan Sulawesi Selatan. Pola usahanya sebagian besar adalah perbibitan atau pembesaran anak, dan hanya sebagian kecil peternak yang mengkhususkan usahanya pada penggemukan ternak (Yusdja et al. 2003).

2.3 Ransum

Ransum merupakan satu atau beberapa jenis bahan pakan yang diberikan untuk seekor ternak selama sehari semalam. Ransum harus dapat memenuhi zat-zat makanan yang dibutuhkan seekor ternak untuk berbagai fungsi tubuhnya, seperti pokok hidup, produksi, maupun reproduksi (Siregar, 1996).

2.4 Algoritme Genetika

Algoritme evolusi mempunyai beberapa cabang algoritme lainnya dan salah satunya adalah algoritme genetika. Algoritme genetika adalah salah satu algoritme dengan teknik pencarian heuristik berdasarkan mekanisme evolusi biologis. Pada umumnya, algoritme genetika digunakan untuk pemecahan masalah yang kompleks (Gen dalam Uyun dan Hartati, 2011).

Tahapan dalam algoritme genetika yaitu inisialisasi kromosom, reproduksi berupa *Crossover* dan mutasi, dan evaluasi. Inisialisasi kromosom berupa gen-gen yang merepresentasikan solusi. Proses penentuan gen dilakukan secara random (*Rianawati & Mahmudy, 2015*). Setelah itu, proses selanjutnya yaitu reproduksi yang melibatkan beberapa parent di dalam *Crossover* dan mutasi. Selanjutnya tahap evaluasi untuk memilih solusi yang akan digunakan pada generasi selanjutnya dengan menggunakan nilai *Fitness.* Evaluasi memiliki banyak jenis seperti *elitism, roulette whel,* dan sebagainya. Hasil algoritme genetika biasanya dalam suatu pemecahan masalah tidak selalu mendapatkan solusi yang terbaik namun mendekati terbaik (*Phetergem dalam devi dkk, 2010*).

Menurut (Michalewis dalam mahmudy, 1996) proses pencarian algoritme genetika berbeda dengan pencarian dan optimasi pada umumnya. Perbedaan tersebut antara lain adalah:

1. Adanya kode dari himpunan parameter dalam manipulasi atau disebut juga kromosom.
2. Proses pencarian dilakukan dari beberapa titik dalam satu populasi, tidak hanya dari satu titik.
3. Menggunakan informasi serta fungsi tujuan pada proses pencarian.
4. Stokastik operator digunakan untuk pencariannya yang bersifat probabilitas, tidak dengan menggunakan aturan deterministik.

2.4.1 Parameter Algoritme Genetika

Berikut adalah parameter yang digunakan dalam algoritme genetika diantaranya adalah:

1. Ukuran populasi *(Popsize)*

   Ukuran populasi menunjukkan jumlah kromosom yang terdapat populasi dalam 1 generasi. Jumlah populasi sangat berpengaruh pada waktu komputasi, jika jumlah populasi banyak, maka waktu yang dibutuhkan untuk memproses data juga akan membutuhkan waktu yang lama, tapi dapat mencegah terjadinya konvergensi. Sedangkan dengan jumlah populasi yang sedikit juga berpengaruh pada solusi yang kurang optimum karena ruang solusi yang dihasilkan kurang mampu merepresentasikan permasalahan yang terkait.

2. *Crossover Rate*

   Crossover Rate atau lebih dikenal dengan C_r adalah perbandingan antara jumlah kromosom dalam populasi dengan jumlah keturunan yang dihasilkan. Tukar silang terjadi saat sebanyak C_r dikalikan dengan *Popsize* dalam populasi. Jika nilai C_r semakin besar, maka alternative solusi yang dihasilkan lebih bervariasi. Sebaliknya, nilai C_r yang kecil, dapat menghasikan solusi yang kurang optimum dikarenakan waktu komputasi yang lama dan proses tukar silang yang banyak.

3. *Mutation Rate*

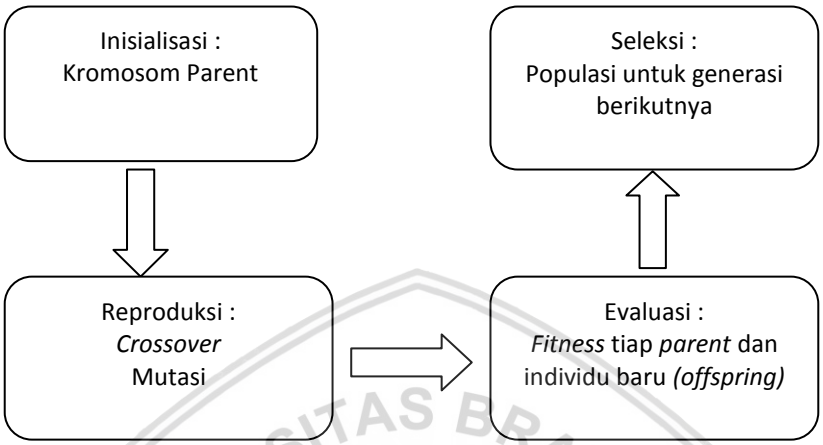
   Mutation Rate atau disebut M_r merupakan parameter yang mempunyai fungsi sebaagai penentu banyaknya kromosom dalam populasi yang akan dimutasi. Jika *Mutation Rate* semakin kecil, maka memungkinkan kromosom tidak melakukan proses mutasi.

4. Jumlah Generasi

   Jumlah generasi ditentukan oleh banyaknya jumlah iterasi yang dilakukan pada proses evaluasi tiap-tiap generasi. Proses dalam menentukan kondisi *break* adalah dengan menentukan jumlah generasi yang tepat agar proses algoritme genetika tidak melakukan perulangan.

2.4.2 Stuktur Algoritme Genetika

Penerapan algoritme genetika memiliki siklus sederhana yang terdiri dari beberapa tahap seperti gambar dibawah ini:



Gambar 2.1 Siklus Algoritme Genetika

Sumber : (Mahmudy, 2013)

A. Inisialisasi

Inisialisasi digunakan untuk menjelaskan nilai dari parameter dalam genetic algorithms yang terdiri dari sejumlah kromosom yang menjadi satu di dalam suatu himpunan atau disebut populasi. Populasi menunjukan sekumpulan individu atau kromosom yang begabung menjadi satu. Ukuran populasi (*Popsize)* adalah banyaknya jumlah kromosom/individu didalam sebuah populasi. Dalam fase ini ditentukan kromosom yang akan menjadi *parent.* Ukuran populasi *(Popsize)* dan harus di tentukan saat proses inisialisasi (*Mamudy*, 2013). Teknik-teknik dalam pembentukan populasi awal antara lain adalah *random search, random generator* dan permutasi gen (Rismayanti & Harihayati, 2015).

B. Representasi kromosom

Representasi kromosom merupakan proses dalam memetakan solusi dari sebuah permasalahan menjadi kromosom. Hasil algoritme genetika dalam memberikan solusi salah satunya dipengaruhi oleh representasi kromosom. Representasi integer, permutasi, real, dan biner adalah beberapa model dalam merepresentasikan sebuah kromosom. Didalam sebuah permasalahan memiliki perbedaan dalam merepresentasikan kromosom karena tidak semua bentuk representasi cocok untuk suatu permasalahan yang ada.

C. Reproduksi

Fase reproduksi ini mempunyai fungsi untuk membentuk individu atau kromosom baru. Terdapat dua fase reproduksi yaitu 11utase dan tukar silang. Berikut penjelasan lebih lengkapnya :

1. Mutasi

Mutasi adalah cara yang digunakan untuk menghasilkan individu baru sebagai anak *(offspring)* sehingga, individu dalam populasi tambah beragam dengan mengubah gen dari keturunan melalui proses acak. *Reciprocal exchange* mutation dan *insertion mutation* adalah 11utase11a 11utase yang biasanya di pakai dalam representasi permutasi. Proses 11utase dari *eciprocal exchange* adalah dengan memilih dua posisi tempat secara random kemudian menukarkan nilai pada posisi keduanya. Dengan hal itu, maka yang akan menghasilkan individu baru. Sedangkan *insertion mutation* dengan memilih gen secara random, selanjutnya gen yang terpilih tersebut diselipkan pada posisi yang dipilih secara acak. Proses 11utase dilakukan setelah proses *Crossover.* Perannya adalah untuk mengganti gen yang hilang dari populasi akibat proses seleksi yang memungkinkan tidak kuncul pada inisialisasi populasi (Kusuman & Purnomo, 2015).

2. Tukar Silang *(Crossover)*

Proses *Crossover* adalah memilih dua *parent* secara random dan terpisah menjadi 2 segmen, sehingga penukaran segmen kromosom *parent* tersebut yang akan menghasilkan keturunan. Parent Setelah terpilih proses selanjutnya melakukan penyilangan untuk menghasilkan anak *(offspring)*. Pada umumnya salah satu metode tukar silang yang sering digunakan adalah *one-cut point Crossover.* Metode ini merepresentasikan bilangan biner dan permutasi dalam menghasilkan individu baru. *Real Code Genetic Algoritms (RCGA)* mempunyai fungsi khusus untuk mengkonversi dari bilangan biner ke bilangan real atau sebaliknya, karena dengan kromosom dalam bentuk bilangan *real* akan membutuhkan waktu yang lama (Mahmudy, 2013). Dalam RCGA *Crossover* yang sering dipakai yaitu *intermediate Crossover* yang mengaplikasikan dengan realcode.

D. Evaluasi

Evaluasi digunakan untuk menghitung *Fitness* setiap kromoso. Dalam menentukan kualitas dari suatu individu pada algoritme genetika nilai *Fitness* yang menjadi sebuah acuan dalam pencapaian nilai paling optimum. Proses ini setiap populasi akan di evaluasi dengan menghitung nilai *Fitness* dari setiap gen dan evaluasi tersebut akan terjadi perulangan sampai kriteria berhenti terpenuhi. Kriteria berhenti antara lain yaitu nilai *Fitness* tidak berubah setekah dalam beberapa generasi secara berturut-turut, kondisi berhenti saat n generasi tidak mendapatkan nilai *Fitness* terbesar, dan *break* pada generasi tertentu. Bila kriteria dalam situasi berhenti dan belum terpenuhi, maka akan terjadi proses pembentukan

individu baru. Jika nilai *Fitness* yang dihasilkan semakin besar maka semakin baik kromosom tersebut untuk menjadi solusi terbaik dalam suatu permasalahan.

E. Seleksi

Fase ini digunakan memilih individu dalam populasi dan individu baru *(offspring)* yang mampu bertahan hidup dan lolos dalam proses seleksi. Pada tahap ini langkah pertama yang dilakukan adalah mencari nilai *Fitness*. Nantinya, pada tahap-tahap selanjutnya nilai *Fitness* ini akan digunakan sebagai acuan. Jika nilai *Fitness* sebuah kromosom semakin besar pula peluang untuk terpilih, sehingga generasi berikutnya akan lebih baik dari generasi sebelumnya. Berikut adalah beberapa metode seleksi yang serig digunakan antara lain seleksi *roulette wheel*, *elitism,* dan *binary tournament*. Metode *roulette wheel* atau disebut juga tochastic sampling with replacement. Pada tahap ini, individu-individu dipetakan dalam suatu segmen secara teratur agar individu memiliki ukuran yang sama dengan ukuran *Fitness*nya. Teknik bilangan acak akan dibangkitkan dan individu yang memiliki segmen dalam kawasan bilangan random tersebut akan diseleksi. Sedangkan *elitism* memungkinkan individu yang bernilai fitness tertinggi akan terpilih. Akan tetapi, kemungkinan individu tersebut akan rusak (nilai fitnessnya menurun) karena terjadinya pindah silang. *Binary Tournament selection* memiliki implementasi yang sederhana dalam algoritme genetika. Dalam tahap ini, n individu dipilih secara acak sebanyak *Popsize*. Individu yang terpilih dibandingkan nilai *Fitness*nya. Nilai *Fitness* yang lebih tinggi akan lolos pada generasi berikutnya.

2.5 Simulated annealing

Simulated annealing adalah algoritme optimasi yang mempunyai sifat *generik*. Dengan berbasis mekanika statistic dan probabilitas, algoritme ini mempunyai kelebihan dalam pencarian pendekatan suatu permasalahan dengan solusi optimum global dan melalui proses *annealing* (pendinginan) yang tediri dari kaca atau baja. Pada umumnya, suatu masalah yang membutuhkan algoritme *simulated annealing* adalah optimasi kombinatorial, seperti halnya tidak memungkinkan solusi optimum untuk sebuah permasalahan dalam ruang pencarian solusi yang begitu kompleks (Mahmudy, 2014). Berikut adalah tabel pemetaan dari *physical annealing* ke *simulated annealing*:

Tabel 2.3 Pemetaan *Physical Annealing* ke *Simulated annealing*

Fisika (terkodinamika)	*Simulated annealing*
Keadaan sistem	Solusi yang pasti
Energi	Biaya

Perubahan Keadaan	Solusi tetangga
Temperatur	Parameter Kontrol
Keadaan Beku	Solusi *Heuristik*

Sumber : (*AlgitowblinkerZ*, 2014)

Penurunan temperatur secara signifikan pada benda padat yang sebelumnya telah dipanaskan hingga mencapai titik beku, kemudian proses pendinginan. *Simulated annealing* menggunakan beberapa parameter yaitu sebagai berikut:

1. Temperatur Awal (T_o) : Memulai iterasi proses *Simulated annealing.* Temperatur awal akan terus berkurang hingga ≤ atau sama dengan nilai temperatur akhir.
2. Temperatur Akhir (T_n) : Penanda batas akhir iterasi.
3. Alpha (α) : Faktor reduksi temperatur sehingga temperatur dapat dikurangi secara bertahap (Cahyaningyas, Ratnawati, & Sutrisno, 2016)

Tahapan pertama pada *simulated annealing* adalah inisialisasi parameter dan individu awal. Teknik inisialisasi yang digunakan sama dengan teknik algoritme genetika untuk populasi awal. Tahapan selanjutnya adalah *neighborhood*.

Modifikasi individu awal menjadi individu baru merupakan proses dari *neighborhood*. Cara modifikasi dengan menukarkan nilai pada salah satu gen ke gen lain pada kromosom yang sama. Kemudian akan melalui proses evalusi terhadap kedua individu. Dari individu awal dan idividu baru nantinya nilai *cost*-nya akan diperbandingkan. Jika selisih (ΔE) berada diantara nilai *cost* individu baru dengan individu awal <= 0, maka individu baru akan terpilih menggantikan individu awal. Sebaliknya, jika selisih dari (ΔE) > 0, maka bilangan random antara 0 hingga 1 akan dibangkitkan. Jika dari probabilitas Boltzman lebih dari bilangan tersebut, maka individu baru akan di terima (Orkcu, 2013). Berikut adalah rumus dari probabilitas Boltzman menggunakan bilangan acak.

$$\text{Exp}(\left(\frac{\Delta E}{T}\right)) \geq \delta \qquad (2.11)$$

Keterangan : Δf = Selisih nilai *cost*

T = Temperatur Sekarang

δ = Bilangan *Random* antara 0-1

Jika nilai random mempunyai nilai lebih besar dibandingkan probabilitas Boltzman, maka akan terjadi penurunan temperatur. Berikut adalah persamaan penurunan temperatur.

$$T_{o+n} = \alpha \times T_s \qquad (2.12)$$

Keterangan : T_{o+n} = Tempertur iterasi berikutnya

α = Alpha

T_s = Temperatur Sekarang

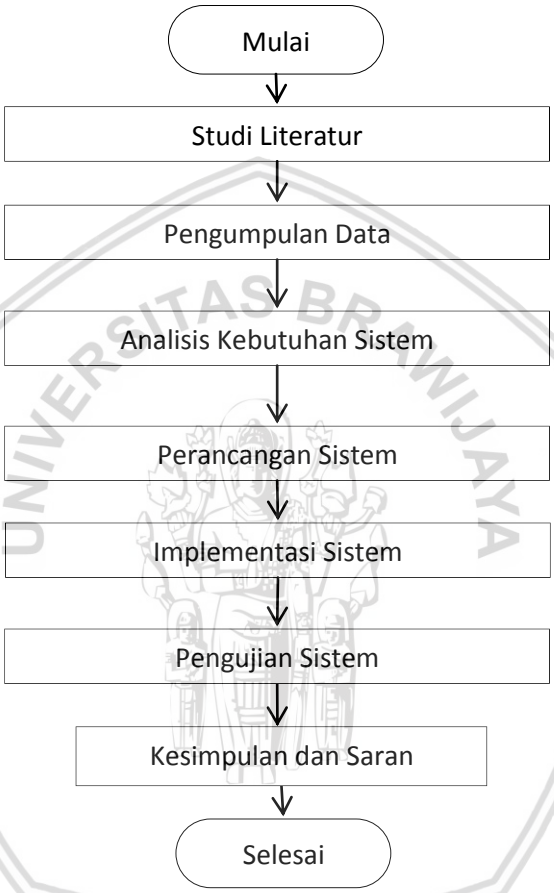
Temperatur akan berhenti jika temperature akhir sudah mencapai akhir.

2.6 Hybrid Algoritme Genetika dan *Simulated annealing*

Kombinasi algoritme genetika - *simulated annealing* atau lebih dikenal dengan GAZA dari beberapa penelitian digunakan dalam masalah optimasi. Kelebihan algoritme genetika terletak pada cara kerja yang pararel. Dengan banyaknya keberagaman individu dalam populasi, memungkinkan algoritme genetika terjebak pada kondisi ekstrim lokal saat bekerja didalam ruang pencarian dan operator-operator genetika tidak dapat menghasilkan *offspring* lebih baik dari *parent*nya (Sofianti, 2004). Dari kelemahan tersebut, *simulated annealing* yang mampu bertahan pada lokal optimum dengan mengendalikan penurunan temperatur suhu sehingga dapat menutupi kelemahan dari algoritme genetika tersebut. Sebaliknya kelemahan *simulated annealing* yang hanya dapat menghasilkan satu solusi saja, sedangkan solusi yang diabaikan tersebut mungkin saja akan lebih baik dari solusi yang dipilih dapat ditutupi dengan kelebihan algoritme genetika. Tujuan penggabungan algoritme genetika - *simulated annealing* (SA) adalah untuk memanfaatkan kelebihan *simulated annealing* yang mampu bertahan menghadapi lokal optimum dan proses pencarian yang dikendalikan oleh suhu untuk digunakan dalam menutupi kekurangan algoritme genetika tersebut.

BAB III METODOLOGI

Secara umum sistem ini akan melakukan optimasi kebutuhan gizi untuk ibu hamil menggunakan algoritme genetika - *simulated annealing*. Hasil optimasi berupa kombinasi bahan makanan dengan kandungan gizi yang seimbang dengan biaya yang minimal setiap harinya. Alur metodologi penelitian ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1 Tahapan Penelitian

Dalam melakukan penelitian terdapat tahapan atau proses-proses utuh yang harus dilakukan. Tahapan-tahapan penelitian akan dilakukan secara lebih rinci akan dijelaskan sebagai berikut:

3.1.1 Studi Literatur

Mempelajari literatur-literatur yang berkaitan dengan pembuatan sistem atau beberapa bidang ilmu lainnya antara lain:

1. Sapi Potong
2. Ransum
3. Algoritme genetika

4. Algoritme *simulated annealing*
5. *Hybridasi* Algoritme genetika dan Algoritme *simulated annealing*

Literatur tersebut diperoleh dari jurnal, ebook, dan penelitian sebelumnya. Hal ini dilakukan untuk dasar referensi terkait dengan tema penelitian.

3.1.2 Pengumpulan Data

Pada tahap pengumpulan pengumpulan data, data yang dibutuhkan adalah data mengenai macam-macam pakan ternak beserta kandungan nutrisi dan harga pakan.Data didapatkan dari Unit Pelayanan Teknis Daerah (UPTD) Kabupaten Malang yang tepatnya berada di kecamatan singosari.

3.1.3 Analisis Kebutuhan Sistem

Tahapan ini dilakukan untuk mengetahui seluruh kebutuhan fungsional dan kebutuhan non fungsional dalam membangun sebuah sistem dengan metode *hybrid* algoritme genetika - *simulated annealing*, sehingga sistem dapat berjalan sesuai dengan tujuan dan harapan. Berikut kebutuhan fungsional sistem :

1. Sistem dapat melakukan proses perhitungan kebutuhan gizi.
2. Sistem dapat melakukan proses optimasi bahan makanan dengan metode *hybrid* algoritme genetika - *simulated annealing.*
3. Sistem mampu menampilkan hasil optimasi kepada pengguna berupa kebutuhan gizi harian berupa bahan pakan untuk memenuhi asupan gizi untuk sapi dan harga masing-masing bahan makanan.

3.1.4 Perancangan Sistem

Tahapan Sistem ini meunjukan proses implementasi terhadap sistem yang akan dikerjakan. Sistem ini meminta inputan dari pengguna berupa nama bahan pakan, nilai gizi dan harga bahan pakan tersebut. Kemudian inputan akan diproses menggunakan algoritme genetika - *simulated annealing* dan menghasilkan *output* berupa kombinasi pakan dengan gizi terbaik dan harga termurah.

3.1.5 Implementasi Sistem

Implementasi sistem menerapkan pada perencanaan sistem yang akan dibuat. Implementasi sistem juga meliputi :

1. Implementasi data menggunakan notepad..
2. Implementasi program menggunakan aplikasi Netbeans.
3. Implementasi algoritme genetika - *simulated annealing* untuk optimasi pakan ternak sapi.

3.1.6 Pengujian Sistem

Pada tahap ini, akan dilakukan penentuan status gizi pakan ternak sapi yang diklasifikasikan sistem dengan metode Hybrid algoritme genetika -

simulated annealing. Adapun parameter-parameter yang akan diuji coba sistem adalah sebagai berikut.

1. Nilai *Popsize*
2. *Crossover Rate (C_r)*
3. *Mutation Rate(M_r)*
4. Jumlah Generasi
5. Temperatur Awal (T_o)
6. Temperatur Akhir (T_n)
7. *Alpha (α)*

3.1.7 Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan bedasarkan hasil setelah beberapa tahap dilakukan antara lain yaitu pengumpulan data, analisis kebutuhan, perancangan, implementasi, dan uji coba sistem dengan menggunakan *hybridasi* metode algoritme genetika - *simulated annealing*. Selain kesimpulan penulisan saran pada penelitian ini dapat digunakan sebagai perbaikan atau pengembangan sistem selanjutnya.

3.2 Algoritme yang Digunakan

Penelitian ini menggunakan *hybridasi* algoritme genetika - *simulated annealing.* Implementasi dilakukan menggunakan bahasa pemrograman Java berbasis GUI karena bersifat *multiplatform* serta menampilkan form yang sederhana dan cukup untuk menampilkan proses hasil perhitungan.

3.3 Kebutuhan Sistem

Terdapat 2 kebutuhan yang digunakan untuk pembuatan sistem ini antara lain :

1. Kebutuhan perangkat keras (Hardware)
2. kebutuhan perangkat lunak (Software).

Hal ini dilakukan untuk mempermudah implementasi, analis, maupun pengujian sistem.

3.3.1 Kebutuhan Perangkat Keras

1. Prosesor : AMD Athlon (tm) II P340 Dual-Core Processor 2.20 GHz
2. RAM : 4.00 GB
3. Monitor : 16 inc
4. Keyboard

3.3.2 Kebutuhan Perangkat Lunak

1. Sistem operasi Windows 10 dengan operating sistemnya 64 bit
2. Netbeans IDE 7.3.1 untuk implementasi (penulisan *source code*) sistem dengan bahasa pemrograman Java.
3. *Java Development Kit* (JDK) sebagai aplikasi pendukung Netbeans
4. *Java Runtime Environtment* (JRE) version 7 update 25 dengan 32 bit
5. Notepad sebagai penyimpan data.

BAB IV PERANCANGAN

Pada bab perancangan ini membahas tentang langkah penyelesaian masalah yang dibahas menggunakan metode yang diajukan, yaitu algoritme genetika - *simulated annealing*. Langkah penyelesaian masalah menggunakan algoritme genetika - *simulated annealing*, perancangan antarmuka serta perancangan uji coba yang akan digunakan dalam pembuatan aplikasi optimasi komposisi pakan sapi menggunakan metode algoritme genetika - *simulated annealing*.

4.1 Perhitungan Kebutuhan Ransum

Ransum adalah campuran pakan yang terdiri lebih dari satu jenis pakan untuk diberikan kepada seekor hewan ternak selama satu hari penuh. Ransum yang diolah harus memenuhi zat-zat makanan yang dibutuhkan oleh seekor binatang ternak dalam memenuhi kebutuhan nutrisi tubuhnya sehari-hari, seperti kebutuhan pokok hidup, produksi dan reproduksinya (Siregar, 1996).

Formulasi ransum bertujuan untuk memberikan pengetahuan dan informasi mengenai beberapa zat makanan serta beberapa bahan makanan yang dapat dijadikan sebuah makanan ternak (ransum) dan tentunya mampu memenuhi kebutuhan hewan ternak agar memperoleh produksi yang sesuai dengan keinginan si peternak (Parakkasi, 1999).

Beberapa langkah-langkah yang harus diketahui dalam penyusunan ransum yaitu sebagai berikut (Indonesia, 2013):

1. Menentukan kebutuhan nutrisi ternak.
2. Menentukan bahan pakan yang ingin digunakan sebagai campuran ransum:
   - Jenis bahan pakan yang disediakan
   - Kandungan nutrisi bahan pakan
   - Harga bahan pakan
3. Memformulasikan berbagai jenis pakan untuk memenuhi kebutuhan hewan ternak dengan teknik perhitungan yang telah ditentukan.
4. Melakukan pengecekan kembali terhadap hasil perhitungan dan kembali disesuaikan berdasarkan kebutuhan nutrisi hewan ternak beserta status fisiologinya.
5. Menyiapkan ransum yang telah sesuai dengan kondisi dan kebutuhan hewan ternak ternak.

Oleh karena itu, hal penting yang perlu dipenuhi dalam proses penyusunan ransum adalah kandungan nutrisi yang harus terkandung didalam pakan tersebut. Hewan ternak harus mengkonsumsi pakan yang didalamnya terkandung semua kebutuhan nutrisinya, namun tetap harus dalam jumlah yang seimbang. Ternak ruminansia dapat tumbuh dengan cepat dan besar, menghasilkan daging, susu dan anak yang banyak serta

sehat apabila pakan yang dikonsumsi sehari-hari mengandung unsur energi, protein, lemak, vitamin, mineral, air dan unsur nutrisi lainnya sesuai dengan status fisiologinya (Indonesia, 2013).

1. Berat Kering

Bahan pakan yang diberikan pada ternak memiliki kadar air yang bervariasi. Bahan kering atau kadar air mengatur konsumsi sehingga menyusun ransum untuk ternak lebih baik dihitung berdasarkan bahan kering(kadar airnya).

2. Kalori (Cal)

Satu Kalori adalah tingkatan panas yang dibutuhkan untuk menaikkan temperatur 1 gram air dari 16,50 C menjadi 17,50 C. Kalori sering digunakan dalam unit pada pakan ternak.

3. Mega Kalori (Mcal)

Satu megakalori sama dengan 1.000.000 kalori dan sering digunakan untuk menggambarkan kebutuhan nutrisi yang lain. Mega kalori berhubungan dengan jumlah energi pakan.

4. TDN (Total Digestible Nutrient)

TDN adalah selisih antara kandungan gizi yang dikonsumsi oleh hewan ternak dengan zat gizi yang terkandung di dalamnya. Faeces merupakan nilai zat gizi yang telah dicerna dan dapat diubah menjadi energi. Hal ini yang membuat nilai TDN dapat dihitung dari konversi DE (digestible atau dimetabolisir, melainkan hanya akan diubah sesuai dengan hukum kekekalan energi). Biasanya ternak ruminansia membutuhkan kandungan serat dalam ransumnya untuk memastikan bahwa rumen sapi berfungsi secara normal. Itu sebabnya TDN dapat diartikan sebagai jumlah energi yang terkandung dalam pakan maupun ransum yang mampu dicerna oleh tubuh.

5. Protein dan Asam Amino

Protein sangat diiperlukan oleh tubuh untuk memperbaiki dan mengganti sel-sel tubuh hewan ternak yang rusak dan ini dapat mempengaruhi hasil produksi dari ternak tersebut. Protein dapat diubah menjadi energi apabila tubuh memerlukannya. Tumbuh-tumbuhan dan biji-bijian menjadi makanan yang mengandung banyak protein sehingga dapat dijadikan makanan pokok untuk hewan ternak. Protein pada tubuh hewan ternak berjenis ruminansia terbagi atas dua macam, yaitu protein yang dapat disintesis dan protein yang tidak dapat disintesis. Hewan ternak jenis ruminansia biasanya memerlukan protein dalam bentuk protein kasar (PK) dan protein yang dapat dicerna. Protein kasar adalah jumlah nitrogen (N) yang ada didalam pakan dikalikan dengan 6,25 (N x 6,25), sedangkan

protein yang dapat dicerna yaitu protein yang diserap dalam saluran pencernaan.

6. Karbohidrat

Karbohirat merupakan sumber energi yang paling besar bagi hewan ternak berjenis ruminansia, karena energi pada pakan berasal dari karbohidrat dan lemak. Sumber karbohidrat mampu didaptkan dari pakan berjenis hijauan dan konsentrat yang diperoleh dari biji-bijian dan limbah hasil pertanian. Biji-bijian semacam jagung, sorgum, gandum dan barley merupakan bahan pakan sumber karbohidrat.

7. Kalsium dan Fosfor

Kalsium dan Fosfor sangat berperan penting dalam untuk pembentukan dan perawatan tulang. Rasio Ca-P untuk hewan ternak berjenis ruminansia disarankan 1:1 sampai 1:2, rasio yang terlalu besar semisal 8:1 mampu menurunkan kemampuan produksi hewan ternak. Komposisi kalsium dan fosfor yang ada pada mineral tubuh berjumlah sebesar 70%. Kalsium berfungsi untuk membentuk perkembangan tulang, proses pembekuan darah, kontraksi otot syaraf, keseimbangan asam-basa dan aktifitas sejumlah enzim.

Untuk memenuhi kebutuhan nutrisi ransum seperti yang dijelaskan sebelumnya, adapun tabel informasi pada Tabel 4.1 dapat membantu dalam penyusunan ransum untuk hewan ternak sapi potong. Hal ini perlu dicantumkan agar campuran ransum sesuai dengan nutrisi yang dibutuhkan hewan ternak berdasarkan jenis pakan dan penambahan berat badan yang diinginkan oleh si peternak. Dan pada Tabel 4.2 terdapat informasi mengenai macam-macam komposisi bahan pakan sapi beserta kandungan nutrisi dan harganya.

Tabel 4. 1 Kebutuhan Nutrisi Sapi Potong Berdasarkan Berat Bobot Badan dan Pertambahan Berat Bobot Badan

Berat Badan (/kg)	PBBH(/kg)	BK(/kg)	ME (Mcal/ kg)	TDN(/kg)	Protein(/gr)	Ca(/gr)	P(/gr)
A. Sapi Jantan							
150	0	3	5,1	1,4	231	6	6
	0,25	3,8	6,56	1,8	400	14	9
	0,5	4,2	8,02	2,2	474	16	10
	0,75	4,4	9,55	2,6	589	21	13
	1	4,5	10,93	3	607	27	16
200	0	3,7	6,3	1,8	285	6	6

Tabel 4. 2 Kebutuhan Nutrisi Sapi Potong Berdasarkan Berat Bobot Badan dan Pertambahan Berat Bobot Badan (lanjutan)

	0,25	4,5	8,1	2,2	470	11	9
	0,5	5,2	9,9	2,8	554	16	12
	0,75	5,4	11,7	3,2	622	21	15
	1	5,6	13,51	3,7	690	27	17
250	0	4,4	7,4	2	337	9	9
	0,25	5,3	9,52	2,6	534	12	10
	0,5	6,2	11,64	3,2	623	16	14
	0,75	6,4	13,78	3,8	693	21	17
	1	6,6	15,84	4,3	760	28	19
Berat Badan (/kg)	PBBH(/kg)	BK(/kg)	ME (Mcal/ kg)	TDN(/kg)	Protein(/gr)	Ca(/gr)	P(/gr)
300	0	5	8,5	2,4	385	10	10
	0,25	6	10,9	3	588	15	11
	0,5	7	13,4	3,7	679	19	14
	0,75	7,4	14,8	4,3	753	23	18
	1	7,5	18,23	5	819	28	21
350	0	5,7	9,5	2,6	432	12	12
	0,25	6,8	12,22	3,3	635	16	14
	0,5	7,9	14,94	4,1	731	20	16
	0,75	8,3	17,66	4,8	806	25	18
	1	8,5	20,38	56	874	30	21
B. Sapi Induk							
-3 Bulan Kebuntingan							
300	0,6	7,40	14,20	3,9	614	18	18
350	0,6	8,30	16,10	4,4	650	19	19
400	0,6	9,20	17,80	4,9	671	19	19
-3 Bulan Terakhir Kebuntingan							
300	0,4	6,90	12,40	3,4	409	11	11
350	0,4	7,70	13,90	3,8	444	12	12

Tabel 4. 3 Kebutuhan Nutrisi Sapi Potong Berdasarkan Berat Bobot Badan dan Pertambahan Berat Bobot Badan (lanjutan)

400	0,4	8,50	15,40	4,2	480	14	14
C. Sapi Menyusui							
300	-	-	15,2	4,2	686	23	23
350	-	-	16,4	4,5	721	24	24
400	-	-	17,5	4,8	757	25	25
D. Sapi Dara							
100	0	2,4	1,1	3,8	93	4	4
	0,25	2,9	1,3	4,9	206	13	10
	0,5	3,1	1,7	6,0	262	14	11
	0,75	3,2	2,0	7,1	319	20	14
	1	3,3	2,3	8,2	375	26	18
Berat Badan (/kg)	PBBH(/kg)	BK(/kg)	ME (Mcal/ kg)	TDN(/kg)	Protein(/gr)	Ca(/gr)	P(/gr)
150	0	3,3	1,6	5,3	127	5	5
	0,25	4,0	1,9	6,8	258	13	11
	0,5	4,2	2,3	8,3	315	14	12
	0,75	4,4	2,7	9,8	368	19	15
	1	4,5	3,1	11,3	428	25	18
200	0	4	1,8	6,5	157	6	6
	0,25	4,9	2,3	8,3	302	10	10
	0,5	5,6	2,8	10,2	358	14	13
	0,75	5,5	3,3	12,1	415	19	16
	1	5,6	3,8	13,9	472	23	18
250	0	4,8	2,1	7,6	185	7	7
	0,25	5,8	2,7	9,8	340	12	12
	0,5	6,2	3,3	12	395	13	13
	0,75	6,5	3,9	14,2	451	18	15
	1	6,6	4,5	16,3	507	23	18
300	0	5,5	2,4	8,8	212	9	9

Tabel 4. 4 Kebutuhan Nutrisi Sapi Potong Berdasarkan Berat Bobot Badan dan Pertambahan Berat Bobot Badan (lanjutan)

	0,25	6,7	3,1	11,2	368	13	13
	0,5	7,1	3,8	13,8	423	14	14
	0,75	7,4	4,5	16,3	502	17	15
	1	7,6	5,2	18,8	535	21	18

Sumber : (Kearl, 1982)

Tabel 4. 5 Komposisi Bahan Pakan

Bahan Pakan	Harga (Rp)	BK(%)	ME (Mcal/kg)	TDN(%)	PK(%)	Ca(%)	P(%)
Bahan Pakan Kelas I							
Jerami (J) Padi Segar	5000	40	1.35	40	4.3	0	0
J. Padi Kering	6000	86	1.27	39	3.7	0	0
Bahan Pakan	Harga (Rp)	BK(%)	ME (Mcal/kg)	TDN(%)	PK(%)	Ca(%)	P(%)
J. jagung bag. Atas Segar	3500	28	2.09	57	8.2	0.54	0.11
Bahan Pakan Kelas II (Hijauan Segar)							
Rumput Gajah	1000	21	1.8	50	8.3	0.59	0.59
Rumput Benggala	3000	27	1.8	50	7.7	0.52	0.22
Rendeng Segar	3000	35	2.45	65	15.1	1.51	0.2
Rendeng Kering	2500	86	1.98	54	14.7	1.5	0.2
Lamtoro Segar	6000	30	2.96	77	23.4	1.4	0.21
Daun Ketela Pohon Segar	5000	26	2.72	71	20	0.99	0.56
Daun Gliciridia Segar	4000	27	2.45	65	19.1	0.67	0.19
Rumput Ilalang	5000	40	1.96	54	5.4	0.13	0.09
Bahan Pakan Kelas IV (Sumber Energi)							
Dedak Halus Padi	3500	86	2.73	70	12.5	0.06	1.55
Dedak Jagung	4000	86	1.85	52	11.3	0.06	0.77
Dedak Gandum	4500	86	2.5	70	15	0.15	1.23

Tabel 4. 6 Komposisi Bahan Pakan (lanjutan)

Jagung Kuning	6000	86	3.12	80	10.3	0.02	0.33
Gaplek	2500	86	2.6	69	1.7	0.1	0.04
Onggok	2000	86	2.45	65	2.2	0.68	0.05
Cantel	7000	86	3.11	80	11.2	0.19	0.2
Tetes	2500	86	1.92	53	4.2	0.71	0.07
Bahan Pakan Kelas V (Sumber Protein)							
Bungkil Kedelai	5500	86	3.02	78	45	0.2	0.74
Bungkil Kacang	3500	86	2.44	65	49.5	0.11	0.74
Bungkil Kelapa	3000	86	2.48	66	21.6	0.47	0.97
Bungkil Kapok	2000	86	2.85	74	31.7	0.22	1.34
Bungkil Kapas	3000	86	2.5	66	44.2	0.31	0.85
Bungkil Kelapa Sawit	2000	86	0	81	20.4	0.31	0.85

Sumber : (Indonesia, 2013)

Catatan : % BK diperoleh dari jumlah berat pakan yang sebenarnya. Komposisi kimiawi yang lainnya dihitung berdasarkan % bahan kering.

Dalam penyusunan ransum khusus untuk melakukan proses penggemukkan sapi, kebutuhan pakan akan dibagi menjadi dua, yaitu pakan tambahan(konsentrat) dan pakan hijauan. Masing-masing bagian tersebut memiliki rasio sekitar 40 : 60. Perbandingan ini sangat sesuai dalam proses penggemukan secara intensif terhadap sapi (Indonesia, 2013). Dan pengukuruan ini berdasarkan kemampuan sapi dalam mengkonsumsi jumlah pakan berdasarkan bobot sapi tersebut. Hal yang dapat dilihat pada Tabel 4.3.

Tabel 4.7 Kemampuan mengkonsumsi jumlah pakan berbobot

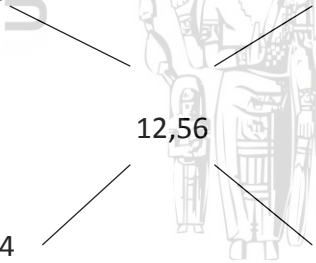
Bobot (Kg)	Kemampuan Mengkonsumsi Pakan (% dari bobot badan)
100 - 150	3,5
150 – 200	4
200 – 250	3,5
250 – 300	3
300 – 350	2,8
350 – 400	2,6
400 – 450	2,4
450 – 500	2

Perkiraan kemampuan konsumsi seperti pada Tabel diatas berdasarkan pakan dengan kandungan keringnya. Berikut ini adalah contoh menyusun ransum untuk seekor sapi jantan dengan bobot 150 kg, target pertambahan berat badan adalah 0,75 kg, maka kebutuhan BK adalah 4,4 kg, TDN = 2,6 kg, PK = 0,589 kg, Ca = 0,021 dan P = 0,013. Bahan pakan yang dipilih dalam penyusuan ransum yaitu rumput gajah, rumput ilalang dan lamtoro segar. Rumput gajah akan digunakan untuk memenuhi kebutuhan bahan kering (BK) 10% dari keseluruhan ransum dikarenakan biayanya yang cukup murah dan memiliki banyak kandungan nutrisi, sehingga bahan kering (BK) rumput gajah adalah (10/100) x 4,4 = 0,44 kg bahan kering.

Kandungan protein rumput gajah = (8,3/100) x 0,44 = 0,03652 kg protein. Untuk menyusun ransum dengan kebutuhan bahan kering (BK) sebesar 4,4 kg dan protein sebesar 0,589 kg masih mengalami kekurangan sehingga perhitungannya sebagai berikut :

BK = 4,4 - 0,44 = 3,96 kg
Protein = 0,589 – 0,03652 = 0,55248 kg, atau
(0,55248/4,4) x 100% = 12.56%.

Kekurangan ini harus dipenuhi dari rumput ilalang dan lamtoro segar dengan perhitungan sebagai berikut :



(17,96/53,92) x 100% = 33,3%

(35,96/53,92) x 100% = 66,7%

Keterangan :

53,96 diperoleh dari hasil penjumlahan 17,96 dengan 35,96, begitu juga dengan 17,96 dihasilkan dari penjumlahan dari 5,4 dengan 12,56. Dan 35,96 diperoleh dari penjumlahan antara 23,4 dengan 12,56.

- Jumlah bahan kering (BK) yang tersedia :
  Rumput Gajah = $\frac{10}{100}$ x 4,4 = 0,44 kg
  Rumput Ilalang = $\frac{33,3}{100}$ x 3,96 = 1,31868 kg
  Lamtoro Segar = $\frac{66,7}{100}$ x 3,96 = 2,6413 kg

- Jumlah kandungan protein kering (PK) yang tersedia :
  Rumput Gajah = $\frac{8,3}{100}$ x 0,44 = 0,03652 kg

Rumput Ilalang = $\frac{5,4}{100}$ x 1,31868 = 0,0712872 kg

Lamtoro Segar = $\frac{23,4}{100}$ x 2,6413 = 0,6180642 kg

- Jumlah kandungan TDN yang tersedia :

Rumput Gajah = $\frac{50}{100}$ x 0,44 = 0,22 kg

Rumput Ilalang = $\frac{54}{100}$ x 1,31868 = 0,712872 kg

Lamtoro Segar = $\frac{77}{100}$ x 2,6413 = 2,033821 kg

- Jumlah kandungan Ca yang tersedia :

Rumput Gajah = $\frac{0,59}{100}$ x 0,44 = 0,002596 kg

Rumput Ilalang = $\frac{0,13}{100}$ x 1,31868 = 0,001714284 kg

Lamtoro Segar = $\frac{1,4}{100}$ x 2,6413 = 0,0369782 kg

- Jumlah kandungan P yang tersedia :

Rumput Gajah = $\frac{0,59}{100}$ x 0,44 = 0,002596 kg

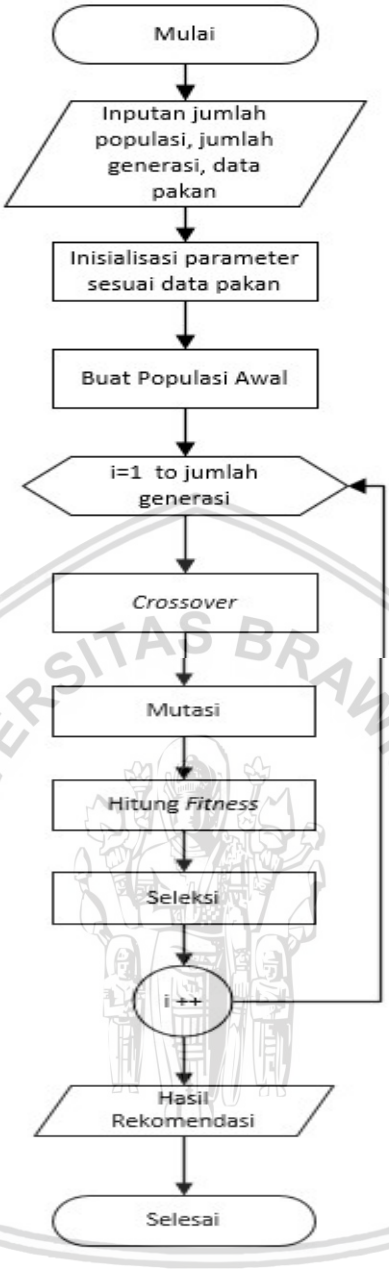
Rumput Ilalang = $\frac{0,09}{100}$ x 1,31868 = 0,001186812 kg

Lamtoro Segar = $\frac{0,21}{100}$ x 2,6413 = 0,00554673 kg

4.2 Penyelesaian Masalah Menggunakan Algoritme Genetika

Untuk menyelesaikan permasalahan optimasi komposisi pakan sapi potong diatas maka adapun tahap-tahapnya akan dijelaskan pada Gambar 4.1.

Parameter yang digunakan pada contoh penelitian ini meliputi: *population size* (nilai yang menyatakan banyaknya individu/kromosom yang ditampung dalam (populasi), *crossover rate* (nilai yang menyatakan rasio *offspring* yang dihasilkan pada proses *crossover* terhadap ukuran populasi sehingga akan dihasilkan *offspring* sebanyak *crossover rate* x *Popsize*) dan kemudian *mutation rate* (nilai yang menyatakan rasio *offspring* yang dihasilkan dari proses mutasi terhadap ukuran populasi sehingga akan dihasilkan *offspring* sebanyak *mutation rate* x *Popsize*). untuk memperoleh komposisi bahan pakan yang memenuhi nutrisi seimbang dan harga minimal, langkah pertama adalah menginputkan berat bobot sapi, bahan pakan yang dipilih serta target pertambahan berat bobot (pbb). Kemudian sistem akan mengambil data kebutuhan nutrisi yang diperlukan dari tabel kebutuhan sapi sesuai berat badan dan target bobot yang telah diinputkan. Setelah mengetahui nutrisi yang dibutuhkan sapi tersebut maka selanjutnya sistem akan melihat kandungan dari setiap bahan pakan, kemudian barulah disusun ransumnya.



Gambar 4.1 *Flowchart* Proses Algoritme Genetika

Setelah mendapatkan berapa banyak kebutuhan makanan sapi, maka tahap berikutnya adalah melakukan inisialisasi awal seperti pada Gambar 4.1. Jumlah bahan pakan yang dipilih oleh user akan menjadi kromosom, kemudian dibangkitkan populasi sebanyak parameter yang dimasukkan. Proses reproduksi dilakukan dengan *crossover* dan mutasi. *Offspring* yang dihasilkan merupakan hasil kali *Popsize* dengan *crossover rate* dan juga hasil kali *Popsize* dengan *mutation rate*. Selanjutnya individu dari populasi awal dan *offspring* hasil *crossover* dan mutasi digabungkan untuk proses seleksi. Metode seleksi yang digunakan adalah seleksi *elitism*. Seleksi dilakukan dengan mengambil sejumlah populasi yang diinputkan dengan

meliha besar *Fitness* nya. Sehingga individu yang memiliki *Fitness* terbesar akan menjadi individu baru untuk generasi selanjutnya.

4.2.1 Representasi Kromosom dan Perhitungan *Fitness*

Representasi kromosom merupakan proses mengubah representasi kasus yang asli dan sebenarnya dari masalah tersebut kedalam bentuk kode tertentu (*Mahmudy*, 2013). Proses pengkodean ini merupakan kunci utama pada persoalan dalam menggunakan algoritme genetika dan erat kaitannya dengan peran kromosom sebagai representasi penyelesaian suatu masalah. Dalam pengkodean masalah kedalam bentuk algoritme genetika sendiri sebenarnya tidak ada aturan khusus yang mengikat, kromosom dapat dibuat dengan kode tertentu dan syarat dari masalah tersebut mampu diproses oleh operator genetika serta merupakan representasi penyelesaian masalah (*Zukhri*, 2014). Representasi kromosom yang digunakan dalam kasus ini adalah *real code* yang dibangkitkan secara acak yang menyatakan inisialisasi berat atau bobot dari pakan yang telah dipilih sebelumnya. Gambar 4.2 merupakan contoh kromosom.

Pakan ke-1	Pakan ke-2	.....	Pakan ke-n
Bobot pakan ke-1	*Bobot pakan ke-2*	....	*Bobot pakan ke-n*

Gambar 4.2 Contoh Representasi Kromosom

Pada gambar diatas menggambarkan bahwa representasi kromosom berupa berat atau bobot dari masing-masing pakan yang ingin digunakan sebagai campuran ransum dengan penjelasan sebagai berikut :

1. Bobot pakan ke-1 merupakan berat dari pakan ke-1 yang akan digunakan dalam bentuk kilogram.
2. Bobot pakan ke-2 merupakan berat dari pakan ke-2 yang akan digunakan dalam bentuk kilogram.
3. n merupakan panjang kromosom atau jumlah jenis pakan yang digunakan.

Adapun contoh pengisian kromosom beserta bobotnya dapat dilihat pada Tabel 4.8 dengan menggunakan pakan Rendeng Kering, Gaplek, Tetes, Dedak Halus Padi.

Tabel 4.8 Contoh Jenis Pakan Beserta Bobotnya

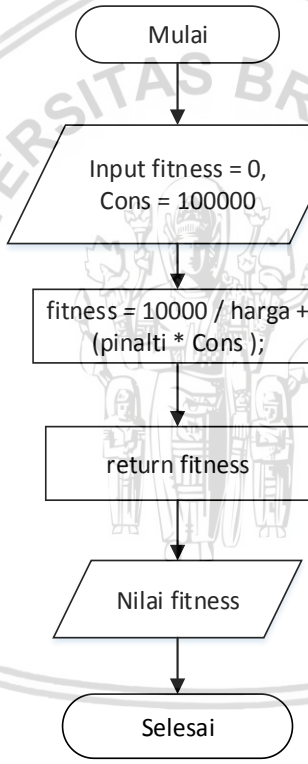
Rendeng Kering	Gaplek	Tetes	Dedak Halus Padi
7.80	1.16	1.73	2.31

Keterangan :

1. Jenis pakan ke-1 adalah Rendeng Kering dengan berat awal sebesar 7,80 kilogram.

2. Jenis pakan ke-2 adalah Gaplek dengan berat awal sebesar 1,16 kilogram.
3. Jenis pakan ke-3 adalah Tetes dengan berat awal sebesar 1,73 kilogram.
4. Jenis pakan ke-4 adalah Dedak Halus Padi dengan berat awal sebesar 2,31 kilogram.

Setiap individu tersebut akan dihitung *Fitness* nya. Fungsi dari nilai *Fitness* sendiri adalah untuk mengetahui seberapa baik suatu individu tersebut dijadikan sebagai solusi (Mahmudy, 2013). Gambar 4.4 merupakan flowchart proses perhitungan nilai *Fitness* yang akan dijelaskan lebih lengkapnya pada persamaan (5-2) beserta cara perhitungan tiap angka penalti per masing-masing nutrisi pada persamaan (4-2) sampai persamaan (4-8).



Gambar 4. 2 *Flowchart* Perhitungan *Fitness*

Untuk mendapatkan nilai *Fitness*, maka perlu diketahui beberapa kebutuhan nutrisi seekor sapi jantan, yaitu kebutuhan BK, TDN, PK, Ca dan P. Berdasarkan dengan bobot badan 150 kg, target pertambahan berat badan 0,75 kg, maka kebutuhan BK adalah 4,4 kg, TDN minimal 2,6 kg, PK minimal 0,589 kg, Ca minimal 0,021, P minimal 0,013, berat kering minimal 5,25 kg, berat pakan hijauan minimal 3,15, berat pakan konsentrat minimal 2,1 kg.

Cara mendapatkan berat kering minimal yaitu berdasarkan Tabel 4.3. Pada Tabel 4.3 diketahui bahwa pada sapi yang mempunyai berat 150 kg dapat mengkonsumsi pakan sebesar 3,5% dari berat badanya,sehingga

apabila dilakukan pengkalkulasian dari 3,5% x 150 didapatkan kemampuan konsumsi berat keringnya sebesar 5,25. Berat pakan hijauan minimal dan berat pakan konsentrat minimal didapatkan dari jumlah perbandingan sebesar 60 : 40. Sehingga berat hijauan diperoleh dari pengkalkulasian 60% x 5,25 = 3,15 kg. Kemudian berat minimalm konsentratnya didapatkan dari pengkalkulasian 40% x 5,25 = 2,1 kg.

Setelah didapatkan informasi mengenai data berat sapi dan penambahan bobot badan sapi tersebut, maka selanjutnya adalah membuat tabel kebutuhan berdasarkan informasi yang didapatkan sebelumnya.

Tabel 4.8 Tabel Kebutuhan

Bahan	Harga (/kg)	Kebutuhan				
		4,4 kg	0,589 kg	2,6 kg	0,021 kg	0,013 kg
		BK(%)	PK(%)	TDN(%)	Ca(%)	P(%)
Rendeng Kering	2500	86	14.7	54	1.5	0.2
Gaplek	2500	86	2.2	69	0.1	0.04
Tetes	2500	86	4.2	53	0.71	0.07
Dedak Halus Padi	3500	86	12.5	70	0.06	1.55

Contoh kromosom pada Tabel 4.5 yang terdiri dari empat gen penyusun yaitu Rendeng Kering, Gaplek, Tetes dan Dedak Halus Padi. Maka proses perhitungan untuk mencari nilai *Fitness* adalah sebagai berikut:

1. Menghitung nilai nutrisi pada setiap individu

Menghitung nilai nutrisi BK, PK, TDN, Ca dan P untuk setiap bahan pakan dengan melihat Tabel 4.5. Rendeng Kering, Gaplek, Tetes dan Dedak Halus Padi masing-masing memiliki kandungan nutrisi BK, PK, TDN, Ca dan P. Tabel 4.6 merupakan tabel nilai nutrisi bk, pk, tdn, Ca dan P untuk setiap bahan yang tersedia.

Tabel 4.9 Nilai BK, PK, TDN, Ca dan P Setiap Bahan Pakan

Bahan	Ketersediaan				
	BK(kg)	PK(gram)	TDN(kg)	Ca(gram)	P(gram)
Rendeng Kering	6.7080	0.9861	3.6223	0.1006	0.0134
Gaplek	0.9938	0.0196	0.7973	0.0012	0.0005
Tetes	1.4907	0.0728	0.9187	0.0123	0.0012
Dedak Halus Padi	1.9876	0.2889	1.6178	0.0014	0.0358

Berikut merupakan contoh perhitungan untuk mendapatkan nilai nutrisi setiap bahan pakan.

Nilai nutrisi Rendeng Kering :

- BK = Bobot Rendeng Kering x (Nutrisi BK Rendeng Kering / 100)
  = $7.8\ x\ (86\ /100)$
  = 6,7080 kg
- PK = Bobot BK Rendeng Kering x (Nutrisi PK Rendeng Kering / 100) x 1000
  = $6{,}708\ x\ (14.7\ /\ 100)$
  = 986,1 gram
- TDN = Bobot BK Rendeng Kering x (Nutrisi TDN Rendeng Kering / 100)
  = $6{,}708\ x\ (54\ /100)$
  = 3,6223 kg
- Ca = Bobot BK Rendeng Kering x (Nutrisi Ca Rendeng Kering / 100) x 1000
  = $6{,}708\ x\ (1.5\ /100)$
  = 100,6 gram
- P = Bobot BK Rendeng Kering x (Nutrisi P Rendeng Kering / 100) x 1000
  = $6{,}708\ x\ (0.2\ /100)$
  = 13,4 gram

2. Menghitung nilai *penalty*

a. *Penalty*

Penalty merupakan nilai yang digunakan pada saat suatu individu melakukan suatu pelanggaran terhadap aturan (Tyas, 2013). Aturan yang dimaksud dalam penelitian ini adalah nutrisi yang digunakan untuk memenuhi kebutuhan pakan sapi berdasarkan anjuran pakar. *Penalty* dihitung pada saat kondisi tertentu yaitu kondisi dimana

kebutuhan nutrisi rekomendasi dari sistem kurang dari kebutuhan nutrisi pakar atau buku. Perhitungan *penalty* ditunjukkan pada persamaan (4-1) (Sari, et al., 2014).

Penalty = *penalty*1 + *penalty*2 + *penalty*3 + *penalty*4 + *penalty*5 + *penalty6* + *penalty7* (4-1)

Dimana :

Penalty1 = nilai *penalty* BK

Penalty2 = nilai *penalty* PK

Penalty3 = nilai *penalty* TDN

Penalty4 = nilai *penalty* Ca

Penalty5 = nilai *penalty* P

Penalty6 = nilai penalty Hijauan

Penalty7 = nilai penalty Konsentrat

Fungsi *penalty* untuk setiap kebutuhan nutrisi ditunjukkan pada persamaan (4-2), (4-3), (4-5), (4-7) dan (4-6) dibawah ini.

$$PenaltyBK = \begin{cases} 0, & TBK\ sistem \geq TBK\ pakar \\ TBK\ pakar - TBK\ sistem, & TBK\ sistem < TBK\ pakar \end{cases} \quad (4\text{-}2)$$

$$PenaltyPK = \begin{cases} 0, & TPK\ sistem \geq TPK\ pakar \\ TPK\ pakar - TPK\ sistem, & TPK\ sistem < TPK\ pakar \end{cases} \quad (4\text{-}3)$$

$$PenaltyTDN = \begin{cases} 0, & TTDN\ sistem \geq TTDN\ pakar \\ TTDN\ pakar - TTDN\ sistem, & TTDN\ sistem < TTDN\ pakar \end{cases} \quad (4\text{-}4)$$

$$PenaltyCa = \begin{cases} 0, & TCa\ sistem \geq TCa\ pakar \\ TCa\ pakar - TCa\ sistem, & TCa\ sistem < TCa\ pakar \end{cases} \quad (4\text{-}5)$$

$$PenaltyP = \begin{cases} 0, & TP\ sistem \geq TP\ pakar \\ TP\ pakar - TP\ sistem, & TP\ sistem < TP\ pakar \end{cases} \quad (4\text{-}6)$$

$$PenaltyBobotHijauan = \begin{cases} 0, & TBH\ sistem \geq TBH\ pakar \\ TBH\ pakar - TBH\ sistem, & TBH\ sistem < TBH\ pakar \end{cases} \quad (4\text{-}7)$$

$$PenaltyBobotKonsentrat = \begin{cases} 0, & TBKons\ sistem \geq TBKons\ pakar \\ TBKons\ pakar - TBKons\ sistem, & TBKons\ sistem < TBKons\ pakar \end{cases} \quad (4\text{-}8)$$

Keterangan :

TBK = Total BK

TPK = Total PK

TTDN = Total TDN

TCa = Total Ca (Kalsium)

TP = Total P (Fosfor)

TBH = Total Berat Hijauan

$$= 60\% \ x \ (Bobot\ Sapi\ x \ \frac{Kemampuan\ Konsumsi}{100})$$

TBKons= Total Berat Konsentrasi

$$= 40\% \ x \ (Bobot\ Sapi\ x \ \frac{Kemampuan\ Konsumsi}{100})$$

Dari Tabel 4.10 maka dapat diketahui nilai nutrisi dari masing-masing bahan pakan. Untuk mendapatkan nilai *penalty*, nilai bobot dikalikan dengan nilai setiap nutrisinya, kemudian dilakukan pemeriksaan ulang dengan kebutuhan sapi, jika kebutuhan nutrisi telah terpenuhi makan nilai penalty nya 0 dan juka ada nutrisi yang tidak terpenuhi akan diperoleh nilai *penalty* nya. Tabel 4.11 merupakan tabel kebutuhan dan ketersediaan nutrisi.

Tabel 4.10 Kebutuhan dan ketersediaan

Keterangan	BK (Kg)	PK (gram)	TDN (Kg)	Ca (gram)	P (gram)	Hijauan (kg)	Konsentrat (kg)	Penalty
Kebutuhan	4,4	0,589	2,6	0,021	0,013	3,15	2,1	0
Keterangan	BK (Kg)	PK (gram)	TDN (Kg)	Ca (gram)	P (gram)	Hijauan (kg)	Konsentrat (kg)	Penalty
Ketersediaan	11,18	1,314	6,49	0,1134	0,0457	6,71	4,47	0

Sehingga dari perhitungan jumlah nutrisi setiap bahan pakan dalam 1 individu maka dapat diketahui nilai *penalty* nya. Berikut merupakan contoh perhitungan untuk medapatkan nilai *penalty*.

- BK = nilai kandungan BK pada Rendeng Kering + nilai kandungan BK pada Gaplek + nilai kandungan BK pada Tetes + nilai kandungan BK pada Dedak Halus Padi
  = 6,7080 + 0,9938 + 1,4907 + 1,9876
  = 11,18 kg
- PK = nilai kandungan PK pada Rendeng Kering + nilai kandungan PK pada Gaplek + nilai kandungan PK pada Tetes + nilai kandungan PK pada Dedak Halus Padi
  = 0,9861 + 0,0169 + 0,0626 + 0,2484
  = 1,3140 kg

- TDN = nilai kandungan TDN pada Rendeng Kering + nilai kandungan TDN pada Gaplek + nilai kandungan TDN pada Tetes + nilai kandungan TDN pada Dedak Halus Padi
  = 3,6223 + 0,6857 + 0,7901 + 1,3913
  = 6,49 kg
- Ca = nilai kandungan Ca pada Rendeng Kering + nilai kandungan Ca pada Gaplek + nilai kandungan Ca pada Tetes + nilai kandungan Ca pada Dedak Halus Padi
  = 0,1006 +0,0010 + 0,0106 + 0,0012
  = 0,1134 kg
- P = nilai kandungan P pada Rendeng Kering + nilai kandungan P pada Gaplek + nilai kandungan P pada Tetes + nilai kandungan P pada Dedak Halus Padi
  = 0,0134 + 0,0004 + 0,0010 + 0,0308
  = 0,0457
- Hijauan = 60% x Total Berat Kering
  = 60% x 11,18
  = 6,71
- Konsentrat = 40% x Total Berat Kering
  = 40% x 11,18
  = 4,47

Karena tidak ada nilai nutrisi yang kurang dari kebutuhan maka nilai Penalty nya adalah 0.

- Dianggap *penalty* jika total bk < kebutuhan bk sapi seperti yang telah dihitung diawal.
- Dianggap *penalty* jika total pk < kebutuhan pk sapi seperti yang telah dihitung diawal.
- Dianggap *penalty* jika total tdn < kebutuhan tdn sapi seperti yang telah dihitung diawal.
- Dianggap *penalty* jika total Ca < kebutuhan Ca sapi seperti yang telah dihitung diawal.
- Dianggap *penalty* jika total P < kebutuhan P sapi seperti yang telah dihitung diawal.
- Dianggap *penalty* jika total berat hijauan pakan < kebutuhan hijauan seperti yang telah dihitung.
- Dianggap *penalty* jika total berat konsentrat pakan < kebutuhan konsentrat seperti yang telah dihitung.

3. Menghitung Harga

Selanjutnya adalah menghitung total harga dalamsatu individu dengan cara:

$$\text{Harga} = \text{Gen pakan } x \text{ harga bahan pakan} \quad (4\text{-}9)$$

Daftar harga setiap bahan pakan dapat dilihat pada halaman lampiran. Tabel 4.12 merupakan tabel total harga setiap individu dalam satu populasi.

Tabel 4. 11 Harga Setiap Individu

Parent	Kromosom				Harga
	R.Kering	Gaplek	Tetes	D.Halus Padi	
P1	7,80	1,16	1,73	2,31	34,811.11
P2	9,00	1,64	2,18	2,18	39,681.82
P3	7,20	1,60	1,60	1,60	31,600.00
P4	9,00	2,00	2,00	2,00	39,500.00
P5	6,60	1,10	1,10	2,20	29,700.00
P6	6,60	1,47	1,96	0,98	28,477.78
P7	6,60	0,73	1,47	2,20	29,700.00

Berikut merupakan contoh perhitungan harga sehingga akan didapatkan harga masing-masing bahan pakan :

- Harga Rendeng Kering = 7,80 x 2500
  = 19500
- Harga Gaplek = 1,16 x 2500
  = 2888.89
- Harga Tetes = 1,73 x 2500
  = 4333.33
- Harga Dedak Padi Halus = 2,31 x 3500
  = 8088.88
- Total = 19500 + 2888.89 + 4333.33 + 8088.88
  = Rp 34811,11

4. Menghitung *Fitness*

Algoritme genetika adalah algortima yang dapat menyelesaikan masalah kompleks, salah satunya yaitu permasalahan optimasi. Dalam permasalahan optimasi harus didapatkan nilai *Fitness*, yaitu nilai dari suatu individu yang menjadi tolak ukur dalam pengevaluasian berdasarkan suatu nilai tertentu. Apabila permasalahannya adalah optimasi untuk memaksimalkan fungsi *h*, maka akan memakai fungsi *f* = *h* dalam mencari nilai *Fitnessnya*. Tapi jika masalah yang ingin diselesaikan berupa optimasi untuk meminimalkan fungsi *h*, maka akan memakai fungsi *f* = *1/h* dalam mendapatkan nilai *Fitnessnya*. Jika *h* bernilai 0 maka akan membuat nilai *f* bernilai tak terhingga. Sehingga nilai *h* pada fungsi *f* = *1/h* harus ditambah dengan bilangan yang dianggap kecil. Sehingga rumus *Fitness* ditunjukkan pada Persamaan (5.1) (Kushardiana, 2013).

$$f = \frac{1}{(h+\alpha)} \qquad (5.1)$$

Dimana :

- α adalah bilangan kecil atau konstanta yang sesuai dengan permasalahan yang dihadapi.
- *h* adalah sebuah fungsi yang akan diminimalkan (dalam kasus ini adalah harga bahan pakan).
- *f* adalah fungsi *Fitness*

Dalam penelitian ini, fungsi yang akan diminimalkan adalah harga dengan menjumlahkannya dengan nilai *Penalty*. Nilai konstanta yang ditetapkan adalah 100000, hal tersebut dikarenakan rentang harga bahan pakan berkisar antara ribuan sampai puluhan ribu, sehingga nilai konstanta ditetapkan menjadi 100000 agar apabila sebuah individu memiliki harga yang murah tetapi memiliki nilai *penalty* tetap memiliki nilai *Fitness* yang jauh lebih kecil dari pada individu yang memiliki harga relatif lebih mahal tetapi tidak memiliki nilai *penalty* sama sekali. Ini bertujuan agar tingkat preferensi atau nilai kepentingan antara harga dan *penalty* sangat diperhatikan, karena seperti apa yang telah dijelaskan sebelumnya bahwa tujuan dari penelitian ini adalah meminimalkan nilai harga dan tetap memperhatikan nilai *penalty*. Oleh karena itu fungsi *Fitness* yang digunakan dalam penelitian ditunjukkan pada Persamaan berikut.

$$f = \frac{10000}{total\ harga + (penalty\ x\ 100000)} \quad (5.2)$$

Dimana :

- Total harga diperolah dengan mengalikan nilai bahan pakan setiap kilogram dengan harga pakan.
- *Penalty* adalah nilai pelanggaran yang dihitung jika kebutuhan nutrisi rekomendasi sistem kurang dari kebutuhan nutrisi oleh pakar dan buku.

Setelah menghitung harga setiap individu, maka langkah berikutnya adalah menghitung nilai *Fitness*. Dengan menggunakan rumus (5.1) maka didapatkan nilai *Fitness* untuk setiap individu yang dapat dilihat pada Tabel 4.13

Tabel 4.12 Nilai *Fitness*

Parent	Kromosom				*Fitness*
	R.Kering	Gaplek	Tetes	D.Halus Padi	
P1	7,80	1,16	1,73	2,31	0,28726460262
P2	9,00	1,64	2,18	2,18	0,25200458190
P3	7,20	1,60	1,60	1,60	0,31645569620
P4	9,00	2,00	2,00	2,00	0,25316455696
P5	6,60	1,10	1,10	2,20	0,33670033670
P6	6,60	1,47	1,96	0,98	0,35115099493
P7	6,60	0,73	1,47	2,20	0,33670033670

Berikut merupakan contoh perhitungan nilai *Fitness* :

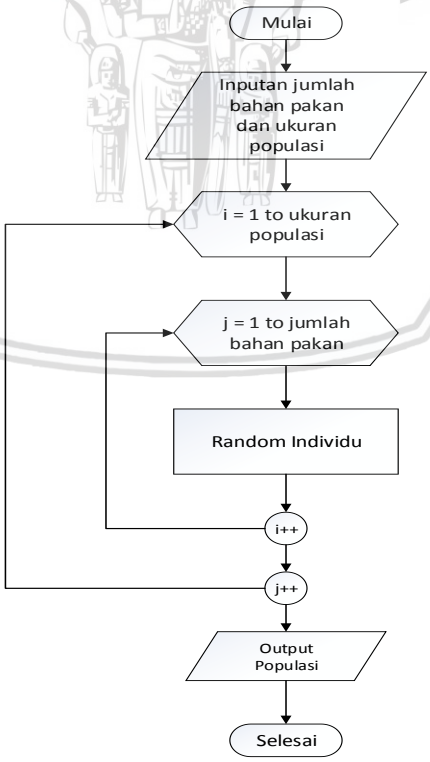
$$f = \frac{10000}{34811 + (0 \times 100000)} = 0{,}28726460262 \quad (5.3)$$

Pada rumus di atas didapatkan nilai *penalty*-nya sama dengan 0. Hal ini dikarenakan pada parent ke-1(P1) memiliki nilai nutrisi yang lebih besar daripada nilai kebutuhan nutrisi sapi yang ingin digemukkan sehingga tidak ada pelanggaran yang didapatkan oleh parent ke-1(P1).

4.2.2 Inisialisasi Populasi Awal

Tahap inisialisasi adalah tahap untuk membangkitkan populasi awal sebanyak jumlah populasi yang sudah ditentukan sebelumnya. Panjang kromosom berjumlah sama dengan jumlah pakan yang diinputkan pada halaman nutrisi pada aplikasi. Jumlah generasi digunakan untuk menentukan berapa banyak ukuran generasi yang diinginkan. Parameter untuk membentuk populasi awal pada algoritme genetika adalah *Popsize, crossover rate, mutation rate* dan generasi.

Popsize bertujuan untuk menyatakan jumlah individu yang mampu ditampung dalam 1 populasi. *Crossover rate* dan *mutation rate* berfungsi sebagai penghasil *offspring* (*child*) sebagai perbandingan untuk mencari nilai *Fitness* terbaik. Pada tahap ini juga menentukan panjang setiap kromosom dengan membangkitkan bilangan acak antara 1-10. Berikut merupakan flowchart dari proses membuat populasi awal dapat dilihat pada Gambar 4.4



Gambar 4.3 Proses Buat Populasi Awal

Untuk menyelesaikan contoh permasalahan di atas maka diperlukan inisialisasi parameter sebagai berikut :

a. Jumlah populasi (*Popsize*) = 7
b. *Crossover rate* (cr) = 0,2
c. *Mutation rate* (mr) = 0,1
d. Jumlah generasi = 1

Dalam contoh kasus ini terdapat 4 bahan yaitu rendeng kering, gaplek, tetes dan dedak halus padi, maka representasi kromosom dapat dilihat pada Tabel 4.10.

Tabel 4.13 Populasi Awal

Parent	Kromosom			
	R.Kering	Gaplek	Tetes	D.Halus Padi
P1	7,80	1,16	1,73	2,31
P2	9,00	1,64	2,18	2,18
P3	7,20	1,60	1,60	1,60
P4	9,00	2,00	2,00	2,00
P5	6,60	1,10	1,10	2,20
P6	6,60	1,47	1,96	0,98
P7	6,60	0,73	1,47	2,20

4.2.3 Reproduksi

Proses selanjutnya adalah proses reproduksi dimana didalam proses ini terdapat 2 proses yaitu *crossover* dan proses mutasi. Proses *crossover* dilakukan dengan metode *extended intermediate crossover* sedangkan proses mutasi menggunakan metode *random mutation*.

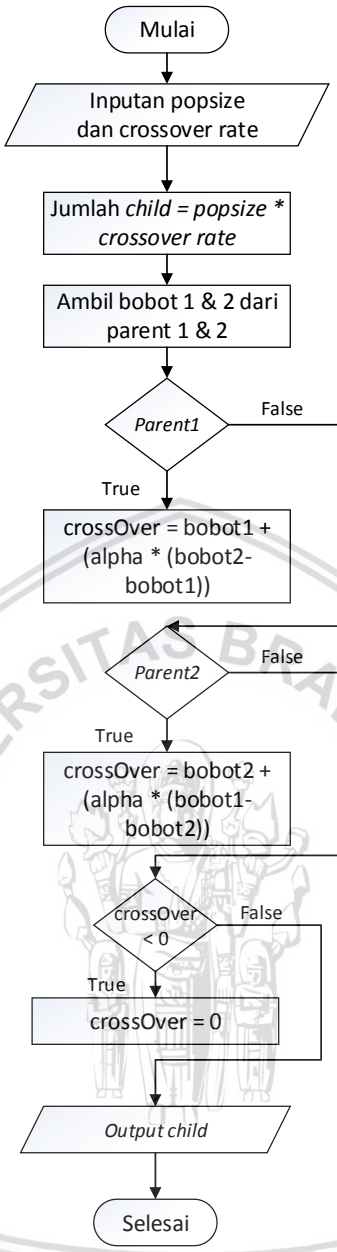
4.2.3.1 Metode Crossover

Metode *crossover* yang digunakan dalam perhitungan ini yaitu *extended intermediate.* Proses crossover akan menghasilkan *offspring* dari hasil kombinasi nilai dua induk yang terpilih secara acak. Jumlah *offspring* yang dihasilkan pada proses *crossover* dapat dilihat berdasarkan *crossover rate* (cr) yang dimasukkan pada program nantinya. Semisal P1 dan P2 adalah dua induk yang terpilih secara acak untuk dilakukan proses *crossover*, maka *offspring* C1 dan C2 dapat dihasilkan dengan Persamaan (5-4) dan Persamaan (5-5) (Mahmudy, 2013). Banyaknya *offspring* yang dibangkitkan pada proses *crossover* adalah hasil perkalian jumlah populasi (*Popsize*) dengan *crossover rate* (cr x *Popsize* = 0,2 x 7 = 1,4 = 2) sedangkan nilai α dipilih secara acak pada interval [0-1] yakni 0,25, 0,40, 0,51 dan 0,92.

$$C1 = P1 + \alpha\,(P2 - P1) \quad (5\text{-}4)$$

$$C2 = P2 + \alpha\,(P1 - P2) \quad (5\text{-}5)$$

Adapun *flowchart* dalam melakukan proses *extended intermadiate crossover* dapat dilihat pada Gambar 4.5



Gambar 4.4 Flowchart proses *crossover*

Proses *crossover* ditunjukkan pada Tabel 4.15 Contoh perhitungan proses crossover dengan induk yang terpilih adalah P1 dan P2 adalah sebagai berikut:

Tabel 4.14 Perhitungan manual *crossover*

Parent	Kromosom				Harga	*Penalty*	*Fitness*
	R.Kering	Gaplek	Tetes	D.Halus Padi			
P1	7,80	1,16	1,73	2,31	34,811.11	0	0,28726460262
P2	9,00	1,64	2,18	2,18	39,681.82	0	0,25200458190
C1	8,10	1,35	1,96	2,19	28,603.56	0	0,34960684290
C2	8,70	1,44	1,95	2,30	30,201.08	0	0,33111402714

Keterangan :

C1 : x1 = 7,80 + 0,25 (9,00 - 7,80) = 8,10
x2 = 1,16 + 0,40 (1,64 - 1,16) = 1,35
x3 = 1,73 + 0,51 (2,18 - 1,73) = 1,96
x4 = 2,31 + 0,92 (2,18 - 2,31) = 2,19

C2 : x1 = 9,00 + 0,25 (7,80 - 9,00) = 8,70
x2 = 1,64 + 0,40 (1,16 - 1,64) = 1,44
x3 = 2,18 + 0,51 (1,73 - 2,18) = 1,95
x4 = 2,18 + 0,92 (2,31- 2,18) = 2,30

4.2.3.2 Perhitungan Mutasi

Mutasi digunakan sebagai operator untuk menjaga keragaman populasi dan untuk membuat individu baru dengan memodifikasi satu atau lebih gen dalam satu individu yang sama (Mahmudy, 2013). Fungsi dari mutasi dalam algoritme genetika adalah untuk menggantikan variasi gen yang hilang selama proses seleksi dalam suatu populasi serta membangkitkan gen yang tidak ada pada populasi awal atau bisa disebut dengan proses eksplorasi. Mutasi akan menambah tingkat variasi individu dalam suatu populasi.

Metode mutasi yang digunakan pada kasus ini yaitu *random mutation.* Proses mutasi ini akan menghasilkan *offspring*(anak) dengan cara menambah ataupun mengurangi nilai gen yang terpilih dengan bilangan *random*. Mutasi dilakukan dengan cara memilih satu *parent*(induk) secara acak, semisal domain variabel xj adalah [min_j, max_j] dan *offspring* yang dihasilkan adalah C = [x'_1.. x'_n] maka nilai gen *offspring* bisa dibangkitkan sebagai berikut :

$$x'_i = x'_i + r(max_i - min_j) \quad (5.5)$$

Metode mutasi ini bekerja dengan cara menambah ataupun mengurangi nilai gen yang terpilih dengan bilangan *random* yang lebih kecil (Mahmudy, 2013) seperti pada persamaan (5.5). Banyaknya *offspring* yang diperoleh pada proses mutasi yaitu dengan cara mengalikan *mutation rate* dengan *Popsize* atau mr x *Popsize* = 0,1 x 7 = 0,7 = 1. Sedangkan nilai *r*

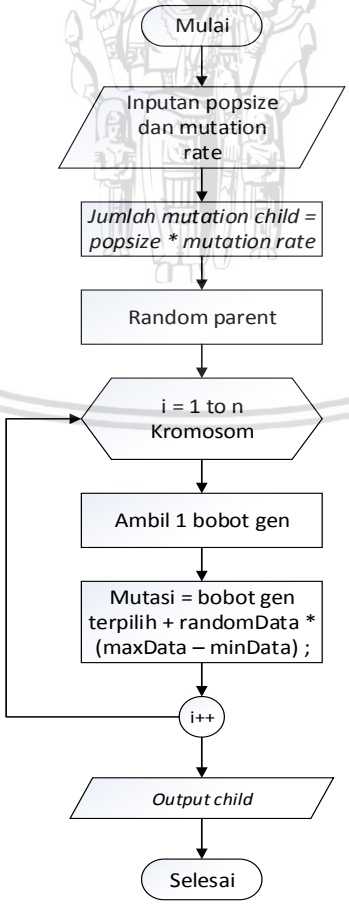
dibangkitkan secara acak dengan interval [-0,2, 0,2] yakni 0,01586 dan individu yang terpilih adalah nomor 2. Maka akan dihasilkan *offspring* 3 pada *flowchart* dalam melakukan proses mutasi dapat dilihat pada Gambar 4.7. Pada Tabel 4.12 contoh perhitungan proses mutasi ddengan induk yang terpilih adalah P2 adalah sebagai berikut :

Tabel 4.15 Perhitungan manual mutasi

Parent	Kromosom				Harga	*Penalty*	*Fitness*
	R.Kering	Gaplek	Tetes	D.Halus Padi			
P3	7,20	1,60	1,60	1,60	31,600.00	0	0,31645569620
C3	7,2	1,620089333	1,6	1,6	31,650.22	0	0,31595353672

Keterangan :

C3 : x1 = 7,2 (tetap)
x2 = 1,6 + (0,01586)*(2 – 0,73) = 1,620089333
x3 = 1,6 (tetap)
x4 = 1,6 (tetap)



Gambar 4.5 *Flowchart* proses mutasi

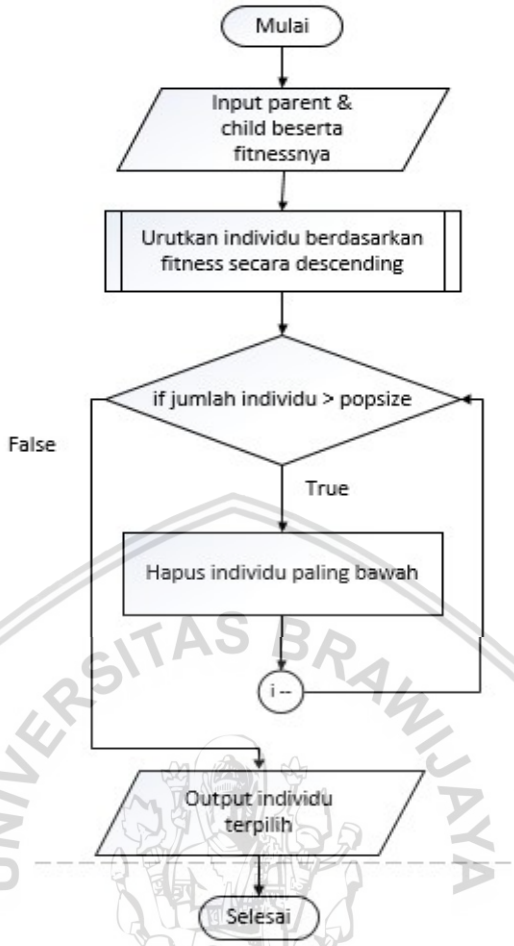
4.2.4 Evaluasi dan Seleksi

Proses evaluasi dilakukan setelah tahap *crossover* dan mutasi telah selesai dilakukan. Proses evaluasi bertujuan untuk menghitung nilai *Fitness* dari setiap kromosom. Proses evaluasi juga bertujuan untuk mengumpulkan seluruh individu yang diantaranya ada *parent* serta *offspring*-nya beserta nilai *Fitness* masing-masing. Semakin besar nilai *Fitness*-nya maka semakin bagus pula kromosom tersebut untuk dijadikan solusi dalam memecahkan suatu masalah (Mahmudy, 2013). Berikut ini adalah tabel hasil perhitungan dalam mencari nilai *Fitness* pada seluruh individu.

Tabel 4.16 Tabel Evaluasi

Parent	Kromosom				Harga	*Penalty*	*Fitness*
	Rendeng Kering	Gaplek	Tetes	Dedak Halus Padi			
P1	7,80	1,16	1,73	2,31	34,811.11	0.000000000	0.28726460262
P2	9,00	1,64	2,18	2,18	39,681.82	0.000000000	0.25200458190
P3	7,20	1,60	1,60	1,60	31,600.00	0.000000000	0.31645569620
P4	9,00	2,00	2,00	2,00	39,500.00	0.000000000	0.25316455696
P5	6,60	1,10	1,10	2,20	29,700.00	0.000000000	0.33670033670
P6	6,60	1,47	1,96	0,98	28,477.78	0.000000000	0.35115099493
P7	6,60	0,73	1,47	2,20	29,700.00	0.000000000	0.33670033670
C1	8,16	1,33	1,94	2,17	36,173.01	0.000000000	0.27644923548
C2	8,64	1,46	1,97	2,33	38,320.25	0.000000000	0.26095861988
C3	7,20	1,62	1,60	1,60	31,650.22	0.000000000	0.31595353672

Seleksi dilakukan untuk memilih individu dan *offspring* yang berada dalam himpunan populasi untuk dipertahankan hidup pada generasi berikutnya (Mahmudy, 2013). Metode seleksi yang digunakan adalah *elitism*, metode seleksi ini bekerja dengan menggabungkan semua individu dalam populasi (*parent*) dan *offspring* dalam satu penampungan (Mahmudy, 2013). Seleksi *elitism* dilakukan dengan memilih *Fitness* paling tinggi, sejumlah dengan jumlah populasi awal. Individu yang mempunyai nilai *Fitness* tinggi akan terpilih menjadi populasi baru digenerasi selanjutnya. Flowchat proses seleksi dengan metode *elitism* dapat dilihat pada Gambar 4.7



Gambar 4.6 Flowchart proses seleksi *elitism*

Hasil seleksi *elitism* dapat dilihat pada Tabel 4.18

Tabel 4.17 Hasil Seleksi Elitism

P(t+1)	Asal P(t)	Kromosom				Harga	Penalty	*Fitness*
		Rendeng Kering	Gaplek	Tetes	D.Halus Padi			
P1	P6	6,60	1,47	1,96	0,98	28,477.78	0	0.351150995
P2	P5	6,60	1,10	1,10	2,20	29,700.00	0	0.336700337
P3	P7	6,60	0,73	1,47	2,20	29,700.00	0	0.336700337
P4	P3	7,20	1,60	1,60	1,60	31,600.00	0	0.316455696
P5	C3	7,20	1,62	1,60	1,60	31,650.22	0	0.315953537
P6	P1	7,80	1,16	1,73	2,31	34,811.11	0	0.287264603
P7	C1	8,16	1,33	1,94	2,17	36,173.01	0	0.276449235

Tabel 4.18 Hasil Seleksi Elitism (lanjutan)

C1	C2	8,64	1,46	1,97	2,33	38,320.25	0	0.26095862
C2	P4	9,00	2,00	2,00	2,00	39,500.00	0	0.253164557
C3	P2	9,00	1,64	2,18	2,18	39,681.82	0	0.252004582

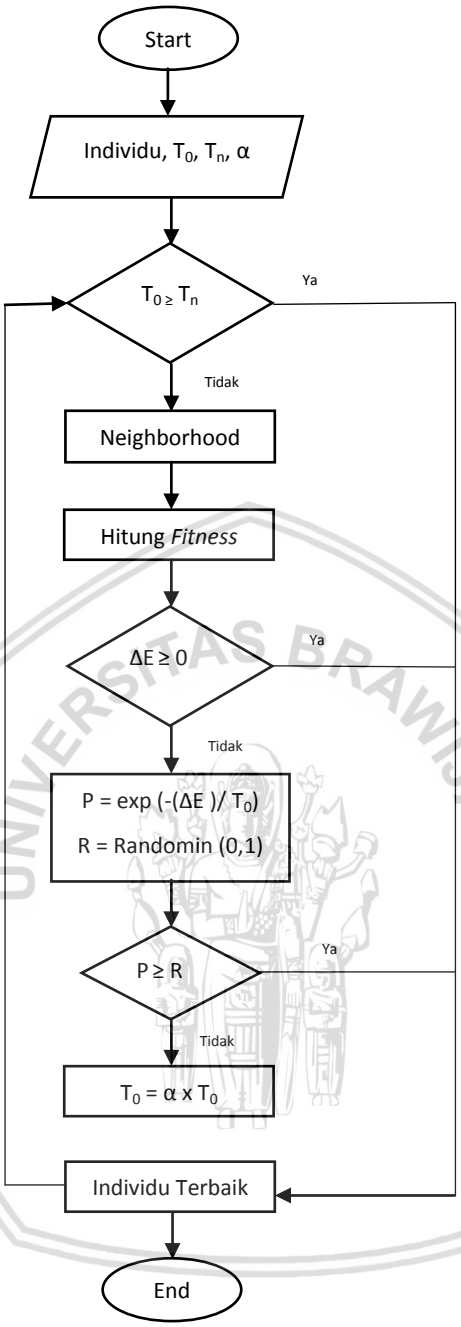
Jadi, komposisi bahan pakan yang terbaik adalah pada *parent* ke 6 dengan bobot 6,6 kg Rendeng Kering, 1,47 kg Gaplek, 1,96 kg Tetes dan 0,98 kg Dedak Halus Padi seharga Rp 28,477,-.

4.3 *Simulated annealing*

Bab ini akan menjelaskan terkait proses penyelesaian permasalahan menggunakan *Simulated annealing*. Proses algoritme simulated annealing. dijalankan setelah proses seleksi pada algoritme genetika selesai dilakukan. Kromosom yang telah melalui seleksi pada algoritme genetika dengan nilai *Fitness* tertinggi digunakan sebagai inisialisasi solusi awal pada algoritme *simulated annealing*.

Berikut adalah langkah-langkah penyelesaian algoritme *simulated annealing*.

1. Inisialisasi nilai temperatur awal (T_0), nilai Alpha (α) dan temperature akhir (T_n).
2. Inisialisasi solusi awal dengan mengambil individu hasil seleksi algoritme genetika dengan *Fitness* terbaik.
3. Melakukan proses neigborhood (memodifikasi kromosom) individu dengan nilai *Fitness* terbaik untuk menghasilkan individu baru dan menghitung juga nilai *Fitness*nya (E. Aycan & T.Ayav, 2009).
4. Melakukan perbandingan, jika selisih nilai *Fitness* antara individu baru dengan individu lama. Jika ΔE lebih dari atau sama dengan 0 maka individu baru tersebut akan mengganti individu lama (Orkcu, 2013).
5. Jika nilai ΔE kurang dari 0 maka proses akan dilanjutkan dengan melakukan perbandingan antara probabilitas Boltzmandengan angka random (Orkcu, 2013).
6. Kemudian jika persamaan 2.6 terpenuhi maka individu baru tersebut akan mengganti individu lama.
7. Jika tidak terpenuhi maka proses akan dilanjutkan dengan melakukan penurunan suhu menggunakan persamaan 2.7 dan melakukan kembali proses perulangan pada langkah pertama.



Gambar 4.7 Diagram Alir *Simulated annealing*

4.3.1 Inisialisasi Parameter

Proses *Simulated annealing* akan dimulai ketika proses algoritme genetika selesai dilakukan. Pada proses ini dibutuhkan tiga parameter yaitu temperatur awal (T_0), temperatur akhir (T_n) dan alpha (α).

1. $T_0 = 0,1$
2. $T_n = 0,9$
3. $\alpha = 0,7$

4.3.2 Neighborhood

Dari inisialisasi parameter *Simulated annealing* Kemudian akan dilakukan proses inisialisasi individu awal yang diambil dari individu terbaik yang dihasilkan oleh proses algoritme genetika. Dari hasil seleksi yang dilakukan diperoleh P6 sebagai individu terbaik. Selanjutnya akan dipilih salah satu gen dari P6 untuk diubah nilainya secara random yang akan menghasilkan offspring baru . Proses neighborhood yang dilakukan dari individu P6 dapat dilihat pada tabel 4.19

Tabel 4.19 Neighboorhood

Individu	Rendeng Kering	Gaplek	Tetes	D. Padi Halus
P6	6,60	1,47	1,96	0,98
C4	6,60	1,47	1,50	0,98

Keterangan :

C4 = offspring dari P6

Perubahan nilai dilakukan pada gen ketiga dari individu p6 dimana yang nilai awal gen ketiga adalah 1,96 diubah menjadi 1,50.

4.3.3 Proses Pengecekan *Fitness*

Dari offspring yang dihasilkan telah didapatkan nilai *Fitness* sebesar 0,34492. Selanjutnya akan dilakukan proses perhitungan selisih antar p6 dengan c4. Jika ΔE lebih dari atau sama dengan 0 maka individu baru dianggap individu terbaik tetapi jika ΔE kurang dari 0 maka proses akan masuk ke perhitungan boltzmann. Proses perhitungan nilai ΔE dapat dilihat pada persamaan 2.6.

$$\Delta E = f(C4) - f(P6) \quad (2.6)$$

$$\Delta E = 0,34492 - 0.35115$$

$$\Delta E = -0,00623$$

4.3.4 Proses Probabilitas Boltzmann dan Penurunan Temperature

Proses ini terjadi apabila ΔE bernilai kurang dari 0, maka proses akan dilanjutkan dengan melakukan proses perbandingan antara probabilitas boltzmann dengan angka random menggunakan persamaan 2.7. Jika probabilitas Boltzman bernilai kurang dari nilai random maka akan masuk ke proses perhitungan temperature baru menggunakan persamaan 2.8, jika temperature baru lebih dari atau sama dengan temperature akhir (T_n) maka proses akan kembali ke proses neighborhood tetapi jika probabilitas boltzmaan bernilai lebih dari nilai random maka offspring dianggap sebagai individu terbaik dan proses berhenti. Penghitungan probabilitas boltzmann dapat dilihat pada persamaan 2.7.

Probabilitas Boltzmann = $\exp(-(-\Delta E)/T_0)$ (2.7)

$= \exp(-(-0{,}00623)/0{,}1)$

$= 1{,}06428$

Nilai random = 0,8

Penghitungan Temperatur baru dapat dilihat pada persamaan 2.8.

$T_0 = \alpha * T_0$ (2.8)

Karena probabilitas Boltzmaan lebih dari Nilai Random maka individu (C4) dianggap sebagai individu terbaik.

BAB V IMPLEMENTASI

Pada bab implementasi ini akan dibahas tentang implementasi sistem yang telah dibuat. Pembahasan implementasi meliputi implementasi program dan implementasi antarmuka.

5.1 Implementasi Program

Pada sub bab ini akan dijelaskan tentang implementasi program berdasarkan perancangan sistem yang sudah dijelaskan pada bab 4. Pada penjelasan implementasi program ini akan dijelaskan proses-proses dan Kode Program setiap fungsi menggunakan bahasa JAVA.

5.1.1 Implementasi Kelas Koneksi

Berikut ini merupakan kode program untuk membuat koneksi dengan database mySQL. Dimana databe yang digunakan bernama db-sapi. Implementasi koneksi database ditunjukkan oleh source code 5.1.

```
    public Connection connect_db(){
            if (koneksi == null){
                try {
                    String url =
"jdbc:mysql://localhost/db_sapi";
                    String user = "root";
                    String pass = "";

Class.forName("com.mysql.jdbc.Driver");
                    koneksi =
DriverManager.getConnection(url,user,pass);
                    stmt =
koneksi.createStatement();
                } catch (Exception exc) {
                    exc.printStackTrace();
                }
            }
            return koneksi;
}
```

Source Code 5.1 Koneksi Data Base

5.1.2 Implementasi Inisialisasi Populasi

Berikut ini merupakan kode program untuk inisialisasi awal populasi yang ditunjukkan oleh source code 5.2.

```
public void Init_Populasi(int Popsize) {
        this.Populasi = new
double[Popsize][4];
        for (int i=0; i < Popsize; i++){
            for (int j=0; j <
this.Populasi[0].length; j++){
                if (j == 0){
                    double nilai_acak = 5.0
+ (10.0 - 5.0) * this.random.nextDouble();
                    this.Populasi[i][j] =
get_nilai_kromosom(nilai_acak);
                } else {
                    double nilai_acak = 0.1
+ (3.0 - 0.1) * this.random.nextDouble();
                    this.Populasi[i][j] =
get_nilai_kromosom(nilai_acak);
                }
            }
        }
    }
```

Source Code 5.2 Inisialisasi Populasi

5.1.3 Implementasi *Crossover*

Berikut ini merupakan kode program untuk *Crossover* yang ditunjukkan oleh source code 5.3.

```
public void Cross_Over(double cr, double[][] Populasi,
double Alpha_Max, double Alpha_Min) {
      int child_crossover = (int) Math.round(cr *
Populasi.length);
      this.Child_CrossOver = new
double[child_crossover][Populasi[0].length];
      double Alpha[] = new double[4];
      for (int i=0; i < Alpha.length; i++){
          Alpha[i] = Alpha_Min + (Alpha_Max -
(Alpha_Min)) * this.random.nextDouble();
      }
```

```
        int counter  = 0;
        this.parent_crossover  = new int[child_crossover][2];
        while (counter < child_crossover){
            int Parent1  = random.nextInt(Populasi.length);
            int Parent2  = random.nextInt(Populasi.length);
            while (Parent1 == Parent2){
                Parent2  = random.nextInt(Populasi.length);
            }
            if (child_crossover - counter == 1){
                for (int j=0; j < Populasi[0].length; j++){
                    this.Child_CrossOver[counter][j]          =
Populasi[Parent1][j] +
                            (Alpha[j]  *  (Populasi[Parent2][j]
- Populasi[Parent1][j]));
                    this.Child_CrossOver[counter][j]          =
get_nilai_kromosom(Child_CrossOver[counter][j]);
                }
                this.parent_crossover[counter][0] = Parent1;
                this.parent_crossover[counter][1] = Parent2;
                counter++;
            } else {
                for (int j=0; j < Populasi[0].length; j++){
                    this.Child_CrossOver[counter][j]          =
Populasi[Parent1][j] +
                            (Alpha[j]  *  (Populasi[Parent2][j]
- Populasi[Parent1][j]));
                    this.Child_CrossOver[counter][j]          =
get_nilai_kromosom(Child_CrossOver[counter][j]);
                }
                this.parent_crossover[counter][0] = Parent1;
                this.parent_crossover[counter][1] = Parent2;
                counter++;
                for (int j=0; j < Populasi[0].length; j++){
                    this.Child_CrossOver[counter][j]          =
Populasi[Parent2][j] +
                            (Alpha[j]  *  (Populasi[Parent1][j]
- Populasi[Parent2][j]));
                    this.Child_CrossOver[counter][j]          =
get_nilai_kromosom(Child_CrossOver[counter][j]);}
                this.parent_crossover[counter][0] = Parent1;
                this.parent_crossover[counter][1] = Parent2;
```

```
                counter++; }}}
```

Source Code 5.3 *Crossover*

5.1.4 Implementasi Mutasi

Berikut ini merupakan kode program untuk mutasi yang ditunjukkan oleh source code 5.4.

```
public void Mutasi(double mr, double[][] Populasi, double Rand_max,
double Rand_min) {
        int   child_mutasi               =   (int)   Math.round(mr   *
Populasi.length);
        this.Child_Mutasi                                    =       new
double[child_mutasi][Populasi[0].length];
        this.parent_mutasi    = new int[child_mutasi];
        int counter = 0;
        while (counter < child_mutasi){
            int Parent1         = random.nextInt(Populasi.length);
            this.parent_mutasi[counter] = Parent1;
            double parent[][]   = ambilparent(Parent1, Populasi);
            int titik_cross     = random.nextInt(parent[0].length);
            double minimum      = getminimum(Populasi, titik_cross);
            double maximum      = getmaximum(Populasi, titik_cross);
            double nilai_acak   = Rand_min + (Rand_max - (Rand_min))
* this.random.nextDouble();
            for (int j=0; j < this.Child_Mutasi[0].length; j++){
                if (j == titik_cross){
                    this.Child_Mutasi[counter][j]  =  parent[0][j]  +
(nilai_acak * (maximum - minimum));
                    this.Child_Mutasi[counter][j]                    =
get_nilai_kromosom(Child_Mutasi[counter][j]);
                } else {
                    this.Child_Mutasi[counter][j] = parent[0][j];
                }
            }
            counter++;
        }
    }
private double getminimum(double[][] Populasi, int titik_cross) {
        double minimum = Populasi[0][titik_cross];
        for (int i=0; i < Populasi.length; i++){
            if (Populasi[i][titik_cross] < minimum){
                minimum = Populasi[i][titik_cross];
            }
        }
        return minimum;
```

```
    }
    private double getmaximum(double[][] Populasi, int titik_cross)
{
        double maximum  = Populasi[0][titik_cross];
        for (int i=0; i < Populasi.length; i++){
             if (Populasi[i][titik_cross] > maximum){
                 maximum = Populasi[i][titik_cross];
             }
         }
        return maximum;
    }
```

Source Code 5.4 Mutasi

5.1.5 Implementasi Gabungan Populasi

Berikut ini merupakan kode program untuk penggabungan seluruh populasi mulai dari populasi awal ditambah dengan populasi hasil crossover dan mutasi yang ditunjukkan oleh source code 5.5.

```
public void set_populasi_gabungan(double Populasi[][], double
child_crossover[][], double child_mutasi[][]){
        int    pop_total                                    =
Populasi.length+child_crossover.length+child_mutasi.length;
        this.Populasi_Gabungan                =           new
double[pop_total][Populasi[0].length];
        int counter_1 = 0;
        int counter_2 = 0;
        for (int i=0; i < Populasi_Gabungan.length; i++){
            if (i < Populasi.length){
                for (int j=0; j < Populasi_Gabungan[0].length; j++){
                    this.Populasi_Gabungan[i][j] = Populasi[i][j];
                }
            } else
                if (i < (Populasi.length+child_crossover.length)){
                    for (int j=0; j < Populasi_Gabungan[0].length;
j++){
                        this.Populasi_Gabungan[i][j]               =
child_crossover[counter_1][j];
                    }
                    counter_1++;
                } else
                    if                        (i                    <
(Populasi.length+child_crossover.length+child_mutasi.length)){
                        for        (int        j=0;        j        <
Populasi_Gabungan[0].length; j++){
                            this.Populasi_Gabungan[i][j]           =
```

```
child_mutasi[counter_2][j];
                        }
                        counter_2++;
                    }
        }
    }
```

Source Code 5.5 Gabungan Populasi

5.1.6 Implementasi Hitung Penalty

Berikut ini merupakan kode program untuk menghitung penalty yang ditunjukkan oleh source code 5.6.

```
        private double hitung_penalty(double[] kebutuhan_sapi, double[][]
        kebutuhan, double berat_hijauan_minimal,
                    double berat_konsentrat_minimal) {
                double nilai_penalty[] = new double[kebutuhan[0].length];
                nilai_penalty = isi(nilai_penalty);
                double hijauan, konsentrat = 0.0;
                double penalty = 0.0;
                for (int i=0; i < kebutuhan.length; i++){
                    for (int j=0; j < kebutuhan[0].length; j++){
                        nilai_penalty[j] += kebutuhan[i][j];
                    }
                }
                hijauan     = (60*nilai_penalty[0])/100;
                konsentrat  = (40*nilai_penalty[0])/100;
                for (int i=0; i < kebutuhan_sapi.length; i++){
                    if (nilai_penalty[i] >= kebutuhan_sapi[i]){
                        penalty += 0.0;
                    } else {
                        penalty += (kebutuhan_sapi[i] - nilai_penalty[i]);
                    }
                }
                if (hijauan >= berat_hijauan_minimal){
                    penalty += 0.0;
                } else {
                    penalty += (berat_hijauan_minimal - hijauan);
                }
                if (konsentrat >= berat_konsentrat_minimal){
                    penalty += 0.0;
                } else {
                    penalty += (berat_konsentrat_minimal - konsentrat);
```

```
            }
            return penalty;
    }
```

Source Code 5.6 Hitung Penalty

5.1.7 Implementasi Hitung Total Harga

Berikut ini merupakan kode program untuk menghitung total harga yang ditunjukkan oleh source code 5.7.

```
public void hitung_total_harga (double Populasi_Gabungan[][], double
kebutuhan_bahan[][]){
        this.total_harga = new double[Populasi_Gabungan.length];
        for (int i=0; i < Populasi_Gabungan.length; i++){
            double tot_harga_temp = 0.0;
            for (int j=0; j < Populasi_Gabungan[0].length; j++){
                tot_harga_temp                                          +=
Populasi_Gabungan[i][j]*kebutuhan_bahan[j][0];
            }
            this.total_harga[i] = tot_harga_temp;
        }
    }
```

Source Code 5.7 Hitung Total

5.1.8 Implementasi Hitung *Fitness*

Berikut ini merupakan kode program untuk menghitung *Fitness* yang ditunjukkan oleh source code 5.8.

```
      public void Fitness(double[][] Populasi_Gabungan, double[] Penalty,
      double[] Total_Harga) {

            this.Fitness = new double [Populasi_Gabungan.length];

            for (int i=0; i < this.Fitness.length; i++){

                this.Fitness[i] = 10000/(Total_Harga[i] + (Penalty[i] *
      100000));
            }
        }
```

Source Code 5.8 Hitung *Fitness*

5.1.9 Implementasi Seleksi

Berikut ini merupakan kode program untuk seleksi yang ditunjukkan oleh source code 5.9.

```
public void Seleksi(double[][] Populasi_Gabungan, double[] Penalty,
double[] Harga, double[] Fitness) {
```

```
        ArrayList<Tampung_Data> list_temp = new ArrayList<>();
        list_temp               = isi_list(Fitness, Penalty);
        list_temp               = urutkan_list(list_temp);
        this.Populasi                                                    =
get_individu_selection(Populasi_Gabungan,list_temp);
        this.Penalty                                                     =
get_penalty_selection(Penalty,list_temp);
        this.total_harga          = get_total_harga_selection(Harga,
list_temp);
        this.Fitness                                                     =
get_Fitness_selection(Fitness,list_temp);
        this.individu_terbaik   = get_individu_terbaik(list_temp);
    }
```

Source Code 5.9 Seleksi

5.1.10 Implementasi Inisialisasi Individu terbaik GA

Berikut ini merupakan kode program untuk inisialisasi individu terbaik yang dihasilkan GA yang ditunjukkan oleh source code 5.10.

```
public void set_solusi_sementara(double populasi[][]){
        this.solusi_sementara = new double[populasi[0].length];

        for (int i=0; i < this.solusi_sementara.length; i++){
            this.solusi_sementara[i] = populasi[0][i];
        }
    }
```

Source Code 5.10 Inisialisasi Individu Terbaik Hasil GA

5.1.11 Implementasi Inisialisasi Individu SA

Berikut ini merupakan kode program untuk Inisialisasi awal *Simulated annealing* yang ditunjukkan oleh source code 5.11.

```
public void modifiikasi_solusi_sementara(int titik){
        if (titik == 0){
            double nilai_acak     = 5.0 + (10.0 - 5.0) *
this.random.nextDouble();
            nilai_acak = get_nilai_kromosom(nilai_acak);
            this.solusi_baru[titik] = nilai_acak;
        } else {
            double nilai_acak     = 0.1 + (3.0 - 0.1) *
this.random.nextDouble();
            nilai_acak = get_nilai_kromosom(nilai_acak);
            this.solusi_baru[titik] = nilai_acak;
        }
```

```
    }
```

Source Code 5.11 Inisialisasi Individu SA

5.1.12 Implementasi Stopping Condition SA

Berikut ini merupakan kode program untuk stopping condition *Simulated annealing* yang ditunjukkan oleh source code 5.12.

```
if (Fitness_solusi_baru > Fitness_solusi_sementara){
                for (int i=0; i < solusi_baru.length; i++){
                    solusi_sementara[i] = solusi_baru[i];
                }
                penalty_solusi_sementara = penalty_solusi_baru;
                total_harga_solusi_sementara = harga_solusi_baru;
                Fitness_solusi_sementara = Fitness_solusi_baru;
            } else {
                double     delta     =     (Fitness_solusi_baru     -
Fitness_solusi_sementara)*-1 ;
                double exp  = Math.exp(delta/T0);
                double rand = random.nextDouble();
                   if (exp > rand){
                       for (int i=0; i < solusi_baru.length; i++){
                           solusi_sementara[i] = solusi_baru[i];
                       }
                       penalty_solusi_sementara                    =
penalty_solusi_baru;
                       total_harga_solusi_sementara                =
harga_solusi_baru;
                       Fitness_solusi_sementara                    =
Fitness_solusi_baru;


                       stop = true;
                   }
            }
             for (int i=0; i < solusi_sementara.length; i++){
                this.TextArea_SA.append("|"+"
"+solusi_sementara[i]+"   "+"|"+"   ");
            }
                this.TextArea_SA.append("\n");
            this.TextArea_SA.append("\n");
            this.TextArea_SA.append("Penalty                        =
"+penalty_solusi_sementara+"       "+"       Total       Harga       =
"+total_harga_solusi_sementara     +"     "+"Fitness     =     "     +
Fitness_solusi_sementara);
            this.TextArea_SA.append("\n");
            if (stop == false){
```

```
                T0 *= Alpha;
            } else {
                break;
            }
        }
```

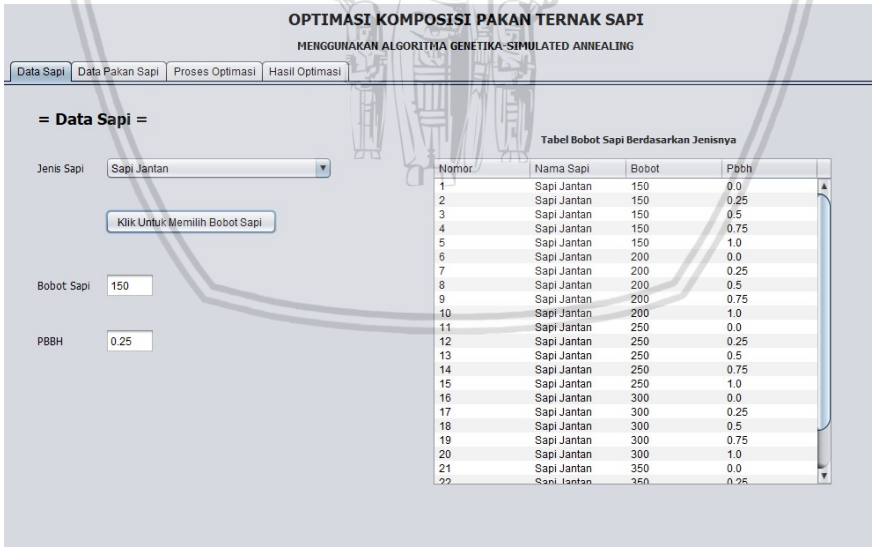
Source Code 5.12 Stopping Condition *Simulated annealing*

5.2 Antarmuka

Pada sub bab ini akan dijelaskan tentang implementasi antarmuka berdasarkan perancangan sistem yang sudah dijelaskan pada bab 4. Implementasi antarmuka terdiri dari 4 halaman yaitu antarmuka halaman untuk memilih jenis sapi beserta bobot dan target pertambahan berat, halaman untuk jenis pakan sapi yang dipilih, halaman untuk proses Algoritme genetika - *simulated annealing* dan halaman hasil akhir dari proses.

5.2.1 Implementasi Antarmuka Halaman Data Sapi

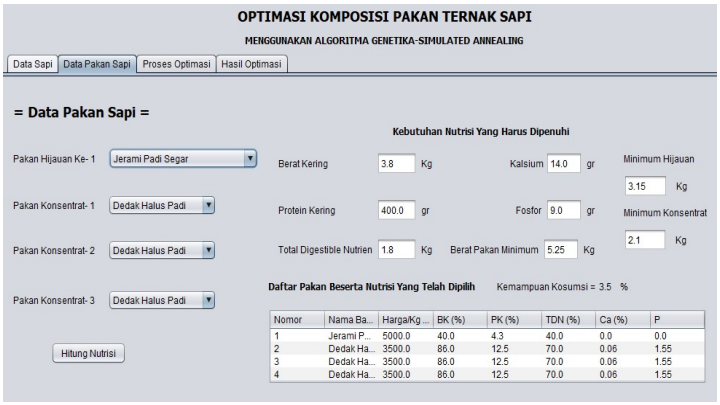
Pada halaman ini akan ditampilkan combobox berupa jenis-jenis sapi, setelah cmemilih jenis sapi kemudian akan dipilih juga bobot sapi dan target pertambahan berat sapi. Implementasi antarmuka halaman data sapi ditunjukkan oleh gambar 5.1.



Gambar 5.1 Antarmuka Halaman Data Sapi

5.2.2 Implementasi Antarmuka Halaman Data Pakan Sapi

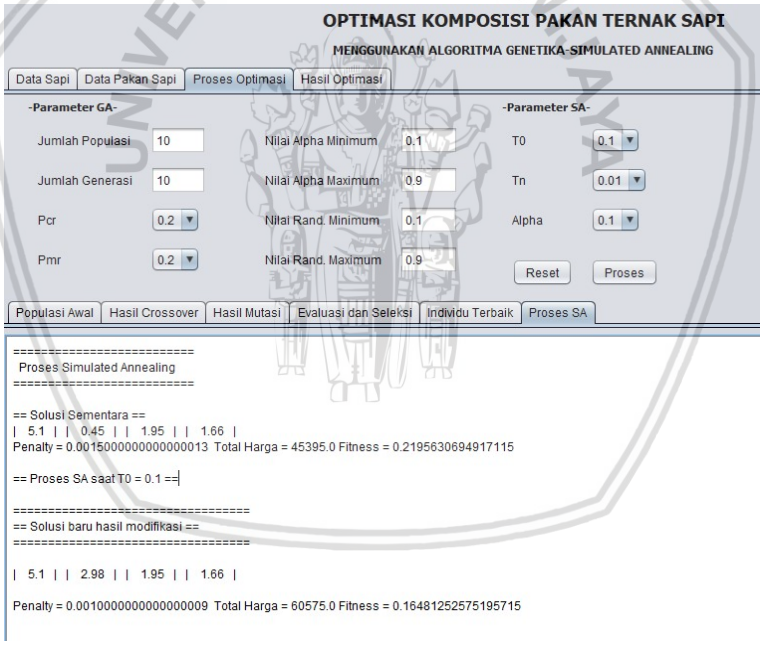
Pada halaman ini akan ditampilkan combobox berupa jenis-jenis pakan sapi, setelah memilih jenis pakan sapi kemudian akan ditampilkan nutrisi yang dibutuhkan oleh sapi. Implementasi antarmuka halaman data pakan sapi ditunjukkan oleh gambar 5.2.



Gambar 5.2 Antarmuka Halaman Data Pakan Sapi

5.2.3 Implementasi Antarmuka Halaman Proses Optimasi

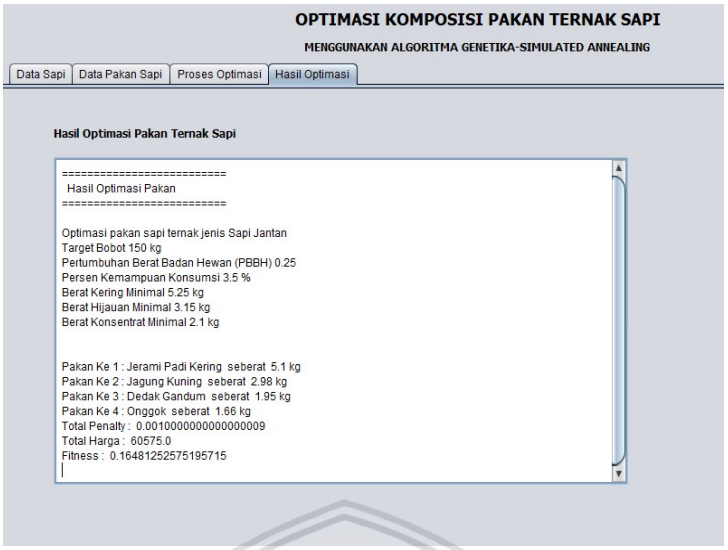
Pada halaman ini akan ditampilkan parameter untuk algoritme genetika - *simulated annealing* yang sebelumnya harus diisi terlebih dahulu oleh pengguna kemudian akan ditampilkan hasil proses dari algoritme genetika - *simulated annealing*. Implementasi antarmuka halaman proses optimasi ditunjukkan oleh gambar 5.3.



Gambar 5.3 Antarmuka Halaman Proses Optimasi

5.2.4 Implementasi Antarmuka Halaman Hasil Optimasi

Pada halaman ini akan ditampilkan hasil proses dari algoritme genetika - *simulated annealing*. Implementasi antarmuka halaman hasil optimasi ditunjukkan oleh gambar 5.4.



Gambar 5.4 Antarmuka Halaman Hasil Optimasi

BAB VI PENGUJIAN & ANALISIS

Pada bab pengujian dan analisis ini menjelaskan tentang pengujian serta analisis terhadap hasil analisis terhadap hasil implementasi yang sudah dirancang pada bab sebelumnya. Pengujian ini dilakukan untuk mengetahui apakah *Fitness* yang dihasilkan akan lebih baik setelah Algoritme Genetika dioptimasi dengan Algoritme *Simulated annealing* ataukah *Fitness* yang dihasilkan justru lebih buruk. Pengujian pada bab ini meliputi pengujian *Fitness* yang dihasilkan oleh Algoritme Genetika dengan *Fitness* yang dihasilkan oleh Algoritme Genetika yang dioptimasi dengan *Simulated annealing*.

6.1 Sistematika Pengujian

Pengujian dilakukan untuk mengetahui apakah *Fitness* yang dihasilkan akan lebih baik setelah Algoritme Genetika dioptimasi dengan Algoritme *Simulated annealing* ataukah *Fitness* yang dihasilkan justru lebih buruk. Oleh karena itu parameter yang digunakan untuk mendapatkan *Fitness* Algoritme Genetika dan *Fitness* Algoritme Genetika yang dioptimasi menggunakan *Simulated annealing* akan selalu diset dengan nilai yang sama sehinggan bisa dibandingkan hasil *Fitness* dari keduanya.

Pengujian ini dilakukan dengan masukan sebagai berikut:

a. Data Sapi

Jenis sapi potong	: Sapi Jantan
Berat bedan	: 300 kg
Penambahan bobot berat badan perhari	: 0,75 kg

b. Data Bahan Pakan

Jenis bahan pakan yang diinputkan dalam setiap proses pengujian adalah rendeng kering, dedak halus padi, dedak jagung dan dedak gandum.

c. Parameter

Crossover Rate = 0.2, Mutation Rate = 0.2 random *Crossover* = 0.1-0.9, random mutation = 0.1-0.9, T0 = 0,9, Tn = 0.1 Alpha = 0.9.

6.2. Analisis dan Pembahasan

Pada pengujian ini parameter yang menjadi pembeda adalah jumlah populasi, jumlah populasi akan diset dengan nilai 10, 20, 30, 40 dan 50, sedangkan untuk setiap jumlah populasi akan dilakukan pengujian dengan set iterasi sebanyak 1-10, untuk setiap jumlah iterasi akan dilakukan pengujian 10 kali kemudian diambil nilai rata-rata nya.

6.2.1 Pengujian Dengan *Popsize* 10

Pengujian yang pertama *Popsize* di set dengan nilai 10 dan iterasi di set dengan nilai 1-10 dimana setiap jumlah iterasi akan dilakukan pengujian 10 kali kemudian diambil nilai rata-rata nya, misal untuk iterasi = 1 dilakukan pengujian sebanyak 10 kali kemudian diambil rata-rata nilai *Fitness*nya, setelah itu dilakukan pengujian untuk iterasi = 2 dilakukan pengujian sebanyak 10 kali kemudian diambil rata-rata nilai *Fitness*nya dan seterusnya hingga iterasi = 10. Hasil Pengujian ditunjukkan oleh tabel 6.1, tabel 6.2, tabel 6.3 dan gambar 6.1.

Tabel 6.1 Hasil Pengujian Algoritme Genetika Untuk *Popsize* = 10

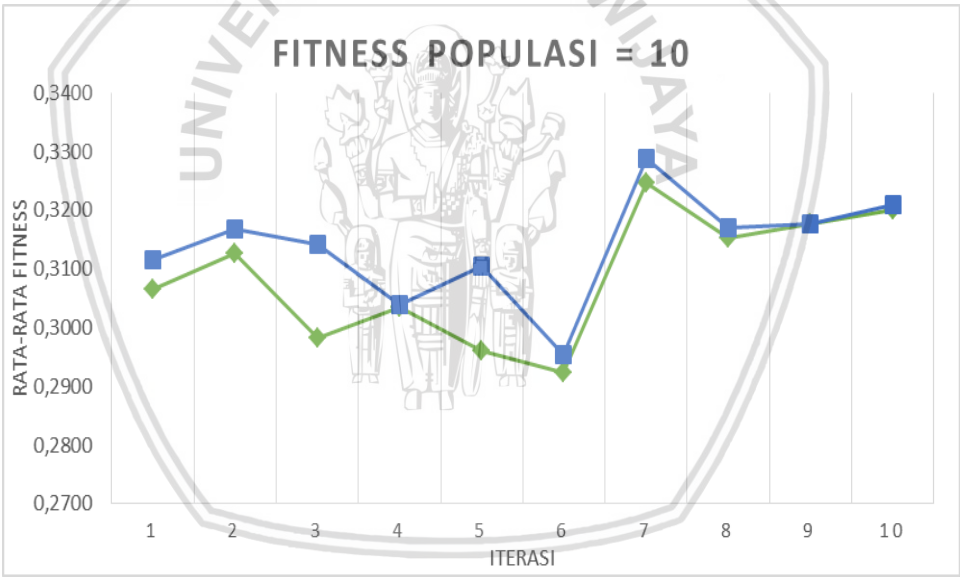
no	Iterasi = 1	Iterasi = 2	Iterasi = 3	Iterasi = 4	Iterasi = 5	Iterasi = 6	Iterasi = 7	Iterasi = 8	Iterasi = 9	Iterasi = 10
1	0,2698	0,3195	0,3204	0,3040	0,3056	0,3034	0,3467	0,3294	0,3448	0,3227
2	0,3113	0,3235	0,2697	0,3066	0,2677	0,3022	0,3393	0,3263	0,3258	0,3222
3	0,2783	0,2952	0,3248	0,3106	0,2963	0,2884	0,3163	0,3067	0,3198	0,3128
4	0,3311	0,3051	0,2953	0,3239	0,3024	0,3048	0,3194	0,3045	0,3338	0,3124
5	0,3266	0,2957	0,2723	0,2990	0,2860	0,2649	0,3252	0,3215	0,2877	0,3254
6	0,2935	0,3153	0,3012	0,2745	0,3197	0,2854	0,3043	0,3132	0,3100	0,3160
7	0,3171	0,3384	0,2935	0,3074	0,2843	0,2825	0,3512	0,3037	0,3339	0,3027
8	0,3115	0,2803	0,3184	0,2987	0,3314	0,2941	0,3239	0,3081	0,3122	0,3320
9	0,3085	0,3263	0,3082	0,3204	0,2755	0,3047	0,3221	0,3301	0,2874	0,3428
10	0,3171	0,3271	0,2775	0,2885	0,2923	0,2923	0,2973	0,3081	0,3218	0,3124

Tabel 6.2 Hasil Pengujian *Simulated annealing* Untuk *Popsize* = 10

no	Iterasi = 1	Iterasi = 2	Iterasi = 3	Iterasi = 4	Iterasi = 5	Iterasi = 6	Iterasi = 7	Iterasi = 8	Iterasi = 9	Iterasi = 10
1	0,2915	0,3223	0,3216	0,3108	0,3073	0,2724	0,3551	0,3313	0,3467	0,3282
2	0,3221	0,3307	0,2887	0,3181	0,2942	0,3045	0,3454	0,3086	0,3364	0,3243
3	0,2854	0,2820	0,3277	0,2869	0,3106	0,3008	0,3168	0,3091	0,3317	0,2973
4	0,3334	0,3212	0,3217	0,3244	0,3502	0,3093	0,3205	0,3016	0,3338	0,3162
5	0,3093	0,2967	0,3193	0,3022	0,2944	0,2765	0,3316	0,3394	0,2595	0,3282
6	0,2994	0,3180	0,3140	0,2817	0,3222	0,2867	0,3046	0,2760	0,3108	0,3199
7	0,3444	0,3463	0,3068	0,3131	0,2934	0,3008	0,3512	0,3328	0,3351	0,3045
8	0,3344	0,2856	0,3332	0,3056	0,3365	0,2847	0,3356	0,3081	0,2809	0,3339
9	0,3141	0,3271	0,3108	0,3229	0,2906	0,3120	0,3231	0,3429	0,3192	0,3428
10	0,2815	0,3378	0,2968	0,2725	0,3044	0,3060	0,3038	0,3210	0,3218	0,3128

Tabel 6.3 Rata-Rata *Fitness* GA dan SA untuk *Popsize* = 10

Iterasi	Rata-Rata *Fitness* GA	Rata-Rata *Fitness* GA-SA
1	0,3065	0,3198
2	0,3126	0,3162
3	0,2981	0,3172
4	0,3034	0,3213
5	0,2961	0,3260
6	0,2923	0,3172
Iterasi	Rata-Rata *Fitness* GA	Rata-Rata *Fitness* GA-SA
7	0,3246	0,3185
8	0,3152	0,3171
9	0,3177	0,3189
10	0,3201	0,3223



Gambar 6.1 Grafik Rata-Rata *Fitness* GA dan GA-SA Untuk *Popsize* = 10

Keterangan:

Garis Biru = Rata – Rata *Fitness* Algoritme Genetika dengan *Simulated annealing*

Garis Hijau = Rata – Rata *Fitness* Algoritme Genetika

Dari grafik yang ditunjukkan oleh gambar 6.1 bisa diambil kesimpulan bahwa untuk jumlah populasi = 10 Algoritme Genetika dengan *Simulated annealing* selalu menghasilkan optimasi yang lebih baik daripada algoritme genetika saja, selain itu keduanya memiliki *Fitness* yang cukup tinggi yang berarti dengan *Popsize* 10 keduanya menghasilkan kombinasi pakan

dengan harga yang murah dan nutrisi yang baik dalam artian nutrisi melebihi kebutuhan sapi.

6.2.2 Pengujian Dengan *Popsize* 20

Pengujian yang pertama *Popsize* di set dengan nilai 20 dan iterasi di set dengan nilai 1-10 dimana setiap jumlah iterasi akan dilakukan pengujian 10 kali kemudian diambil nilai rata-rata nya, misal untuk iterasi = 1 dilakukan pengujian sebanyak 10 kali kemudian diambil rata-rata nilai *Fitness*nya, setelah itu dilakukan pengujian untuk iterasi = 2 dilakukan pengujian sebanyak 10 kali kemudian diambil rata-rata nilai *Fitness*nya dan seterusnya hingga iterasi = 10. Hasil Pengujian ditunjukkan oleh tabel 6.4, tabel 6.5, tabel 6.6 dan gambar 6.2.

Tabel 6.4 Hasil Pengujian Algoritme Genetika Untuk *Popsize* = 20

no	Iterasi = 1	Iterasi = 2	Iterasi = 3	Iterasi = 4	Iterasi = 5	Iterasi = 6	Iterasi = 7	Iterasi = 8	Iterasi = 9	Iterasi = 10
1	0,3273	0,3094	0,3438	0,3456	0,3167	0,3105	0,3263	0,3212	0,3008	0,3307
2	0,3216	0,3229	0,3162	0,3166	0,3175	0,3217	0,3348	0,3128	0,3128	0,3077
3	0,3276	0,3401	0,3051	0,3266	0,3328	0,3147	0,3053	0,3088	0,3345	0,3271
4	0,3226	0,3264	0,2977	0,3266	0,3274	0,3352	0,3174	0,3166	0,3125	0,3303
5	0,3065	0,2904	0,2970	0,3165	0,3405	0,3203	0,3195	0,3189	0,3154	0,3356
6	0,3128	0,3092	0,3424	0,3274	0,3135	0,3073	0,3322	0,3326	0,3326	0,3163
7	0,3246	0,3206	0,3266	0,3111	0,3127	0,3006	0,2972	0,3100	0,3125	0,3067
8	0,3195	0,3110	0,3352	0,3334	0,3509	0,3210	0,3124	0,3097	0,3194	0,3183
9	0,3333	0,3075	0,3126	0,2942	0,3174	0,3190	0,3205	0,3132	0,3220	0,3218
10	0,3025	0,3245	0,2956	0,3154	0,3305	0,3216	0,3190	0,3265	0,3265	0,3281

Tabel 6.5 Hasil Pengujian *Simulated annealing* Untuk *Popsize* = 20

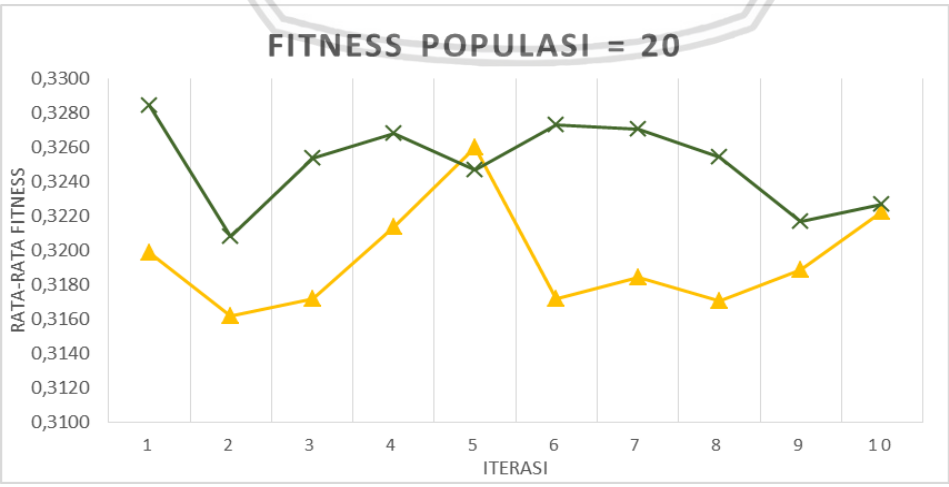
no	Iterasi = 1	Iterasi = 2	Iterasi = 3	Iterasi = 4	Iterasi = 5	Iterasi = 6	Iterasi = 7	Iterasi = 8	Iterasi = 9	Iterasi = 10
1	0,3352	0,3140	0,3453	0,3480	0,3412	0,3475	0,3281	0,3255	0,3089	0,3318
2	0,3216	0,3237	0,3248	0,3282	0,3204	0,3324	0,3561	0,3206	0,3182	0,3175
3	0,3364	0,3401	0,3118	0,3355	0,3350	0,3215	0,3140	0,3143	0,3388	0,3458
4	0,3260	0,3282	0,3331	0,3325	0,3441	0,3360	0,3291	0,3298	0,3225	0,2637
5	0,3466	0,2946	0,3098	0,2853	0,2972	0,3385	0,3241	0,3212	0,3182	0,3361
6	0,3283	0,3108	0,3424	0,3293	0,3167	0,3138	0,3456	0,3338	0,3351	0,3249

Tabel 6.6 Hasil Pengujian *Simulated annealing* Untuk *Popsize* = 20 (lanjutan)

7	0,3284	0,3261	0,3382	0,3483	0,2705	0,3118	0,2979	0,3120	0,3137	0,3168
8	0,3224	0,3219	0,3368	0,3452	0,3512	0,3253	0,3141	0,3346	0,3197	0,3289
9	0,3345	0,3160	0,3154	0,2955	0,3371	0,3192	0,3333	0,3176	0,3125	0,3229
10	0,3048	0,3328	0,2959	0,3204	0,3332	0,3267	0,3282	0,3448	0,3294	0,3381

Tabel 6.7 Rata-Rata *Fitness* GA dan SA untuk *Popsize* = 20

Iterasi	Rata-Rata *Fitness* GA	Rata-Rata *Fitness* GA-SA
1	0,3198	0,3284
2	0,3162	0,3208
3	0,3172	0,3253
4	0,3213	0,3268
5	0,3260	0,3246
Iterasi	Rata-Rata *Fitness* GA	Rata-Rata *Fitness* GA-SA
6	0,3172	0,3273
7	0,3185	0,3271
8	0,3171	0,3254
9	0,3189	0,3217
10	0,3223	0,3227



Gambar 6.2 Grafik Rata-Rata *Fitness* GA dan GA-SA Untuk *Popsize* = 20

Keterangan:

Garis Hijau = Rata – Rata *Fitness* Algoritme Genetika dengan *Simulated annealing*
Garis Kuning = Rata – Rata *Fitness* Algoritme Genetika

Dari grafik yang ditunjukkan oleh gambar 6.2 bisa diambil kesimpulan bahwa untuk jumlah populasi = 20 Algoritme Genetika dengan *Simulated annealing* 9 kali menghasilkan optimasi yang lebih baik daripada algoritme genetika saja, selain itu keduanya memiliki *Fitness* yang cukup tinggi yang berarti dengan *Popsize* 20 keduanya menghasilkan kombinasi pakan dengan harga yang murah dan nutrisi yang baik dalam artian nutrisi melebihi kebutuhan sapi.

6.2.3 Pengujian Dengan *Popsize* 30

Pengujian yang pertama *Popsize* di set dengan nilai 30 dan iterasi di set dengan nilai 1-10 dimana setiap jumlah iterasi akan dilakukan pengujian 10 kali kemudian diambil nilai rata-rata nya, misal untuk iterasi = 1 dilakukan pengujian sebanyak 10 kali kemudian diambil rata-rata nilai *Fitness*nya, setelah itu dilakukan pengujian untuk iterasi = 2 dilakukan pengujian sebanyak 10 kali kemudian diambil rata-rata nilai *Fitness*nya dan seterusnya hingga iterasi = 10. Hasil Pengujian ditunjukkan oleh tabel 6.7, tabel 6.8, tabel 6.9 dan gambar 6.3.

Tabel 6.8 Hasil Pengujian Algoritme Genetika Untuk *Popsize* = 30

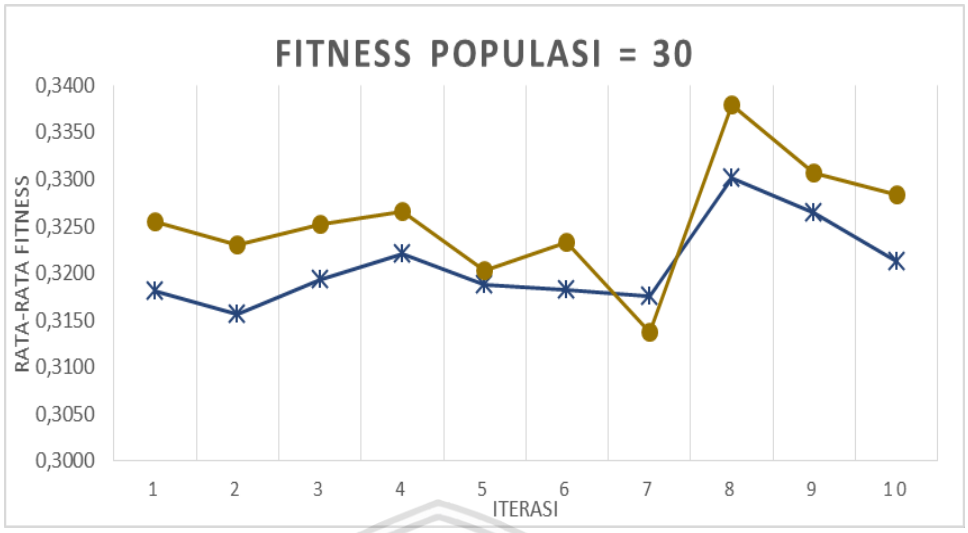
no	Iterasi = 1	Iterasi = 2	Iterasi = 3	Iterasi = 4	Iterasi = 5	Iterasi = 6	Iterasi = 7	Iterasi = 8	Iterasi = 9	Iterasi = 10
1	0,3319	0,3259	0,3042	0,3159	0,3128	0,3316	0,3225	0,3422	0,3086	0,3188
2	0,3087	0,3097	0,3255	0,3377	0,3024	0,3050	0,3203	0,3332	0,3271	0,3329
3	0,3295	0,3170	0,3318	0,3210	0,3117	0,3390	0,3130	0,3259	0,2964	0,3289
4	0,3176	0,3065	0,3258	0,3042	0,3291	0,3152	0,3134	0,3409	0,3479	0,3206
5	0,3018	0,3229	0,3099	0,3171	0,3598	0,3079	0,3280	0,3263	0,3155	0,3275
6	0,3165	0,3264	0,3183	0,3430	0,3191	0,3104	0,3283	0,3085	0,3325	0,3165
7	0,3294	0,3155	0,3074	0,3220	0,2930	0,3107	0,3108	0,3179	0,3378	0,3169
8	0,3034	0,3172	0,3216	0,3236	0,3234	0,3182	0,3005	0,3348	0,3514	0,3138
9	0,3131	0,3036	0,3376	0,3112	0,3224	0,3326	0,3308	0,3239	0,3240	0,3121
10	0,3294	0,3117	0,3116	0,3248	0,3136	0,3124	0,3084	0,3475	0,3229	0,3249

Tabel 6.9 Hasil Pengujian *Simulated annealing* Untuk *Popsize* = 30

no	Iterasi = 1	Iterasi = 2	Iterasi = 3	Iterasi = 4	Iterasi = 5	Iterasi = 6	Iterasi = 7	Iterasi = 8	Iterasi = 9	Iterasi = 10
1	0,3324	0,3324	0,3056	0,2641	0,3210	0,3380	0,3233	0,3450	0,3127	0,3357
2	0,3273	0,3285	0,3269	0,3414	0,3070	0,3204	0,3210	0,3374	0,3350	0,3401
3	0,3318	0,3179	0,3337	0,3246	0,3162	0,3040	0,2701	0,3320	0,3131	0,3359
4	0,3368	0,3082	0,3315	0,3161	0,3437	0,3200	0,3152	0,3474	0,3484	0,3246
5	0,3063	0,3273	0,3179	0,3254	0,3038	0,3102	0,3291	0,3324	0,3197	0,3309
6	0,3192	0,3303	0,3245	0,3442	0,3222	0,3278	0,2750	0,3321	0,3393	0,3210
7	0,3269	0,3190	0,3131	0,3249	0,2933	0,3146	0,3400	0,3228	0,3452	0,3272
8	0,3096	0,3452	0,3332	0,3519	0,3478	0,3269	0,3009	0,3366	0,3594	0,3174
9	0,3336	0,3053	0,3396	0,3391	0,3235	0,3326	0,3398	0,3322	0,3292	0,3187
10	0,3313	0,3162	0,3266	0,3336	0,3241	0,3383	0,3232	0,3611	0,3052	0,3317

Tabel 6.10 Rata-Rata *Fitness* GA dan SA untuk *Popsize* = 30

Iterasi	Rata-Rata *Fitness* GA	Rata-Rata *Fitness* GA-SA
1	0,3181	0,3255
2	0,3156	0,3230
3	0,3194	0,3253
4	0,3221	0,3265
5	0,3187	0,3202
Iterasi	Rata-Rata *Fitness* GA	Rata-Rata *Fitness* GA-SA
6	0,3183	0,3233
7	0,3176	0,3185
8	0,3301	0,3171
9	0,3264	0,3189
10	0,3213	0,3223



Gambar 6.3 Grafik Rata-Rata *Fitness* GA dan GA-SA Untuk *Popsize* = 30

Keterangan :

Garis kuning = Rata – Rata *Fitness* Algoritme Genetika dengan *Simulated annealing*

Garis Biru = Rata – Rata *Fitness* Algoritme Genetika

Dari grafik yang ditunjukkan oleh gambar 6.3 bisa diambil kesimpulan bahwa untuk jumlah populasi = 30 Algoritme Genetika dengan *Simulated annealing* 9 kali menghasilkan optimasi yang lebih baik daripada algoritme genetika saja, selain itu keduanya memiliki *Fitness* yang cukup tinggi yang berarti dengan *Popsize* 30 keduanya menghasilkan kombinasi pakan dengan harga yang murah dan nutrisi yang baik dalam artian nutrisi melebihi kebutuhan sapi.

6.2.4 Pengujian Dengan *Popsize* 40

Pengujian yang pertama *Popsize* di set dengan nilai 40 dan iterasi di set dengan nilai 1-10 dimana setiap jumlah iterasi akan dilakukan pengujian 10 kali kemudian diambil nilai rata-rata nya, misal untuk iterasi = 1 dilakukan pengujian sebanyak 10 kali kemudian diambil rata-rata nilai *Fitness*nya, setelah itu dilakukan pengujian untuk iterasi = 2 dilakukan pengujian sebanyak 10 kali kemudian diambil rata-rata nilai *Fitness*nya dan seterusnya hingga iterasi = 10. Hasil Pengujian ditunjukkan oleh tabel 6.10, tabel 6.11, tabel 6.12 dan gambar 6.4.

Tabel 6.11 Hasil Pengujian Algoritme Genetika Untuk *Popsize* = 40

no	Iterasi = 1	Iterasi = 2	Iterasi = 3	Iterasi = 4	Iterasi = 5	Iterasi = 6	Iterasi = 7	Iterasi = 8	Iterasi = 9	Iterasi = 10
1	0,3698	0,3164	0,3507	0,3318	0,3158	0,3118	0,3333	0,3544	0,3294	0,3459

Tabel 6.12 Hasil Pengujian Algoritme Genetika Untuk *Popsize* = 40 (lanjutan)

2	0,3216	0,3172	0,3150	0,3165	0,3198	0,3188	0,3051	0,3333	0,3122	0,3180
3	0,3074	0,3214	0,3142	0,3226	0,3187	0,3537	0,3356	0,3181	0,3250	0,3218
4	0,3338	0,3316	0,3301	0,3249	0,3203	0,3138	0,3221	0,3364	0,3318	0,3209
5	0,3160	0,3219	0,3102	0,3277	0,3348	0,3136	0,3258	0,3335	0,3131	0,3234
6	0,3140	0,3173	0,3360	0,3231	0,3136	0,3268	0,3592	0,3119	0,3175	0,3388
7	0,3155	0,3122	0,3182	0,3452	0,3258	0,3633	0,3251	0,3271	0,3340	0,3388
8	0,3274	0,3238	0,3179	0,3373	0,2993	0,3475	0,3364	0,3269	0,3284	0,3342
9	0,3271	0,3125	0,3018	0,3106	0,3214	0,3371	0,3225	0,3307	0,3173	0,3239
10	0,3441	0,3141	0,3204	0,3045	0,3429	0,3297	0,3137	0,3219	0,3421	0,3234

Tabel 6.13 Hasil Pengujian *Simulated annealing* Untuk *Popsize* = 40

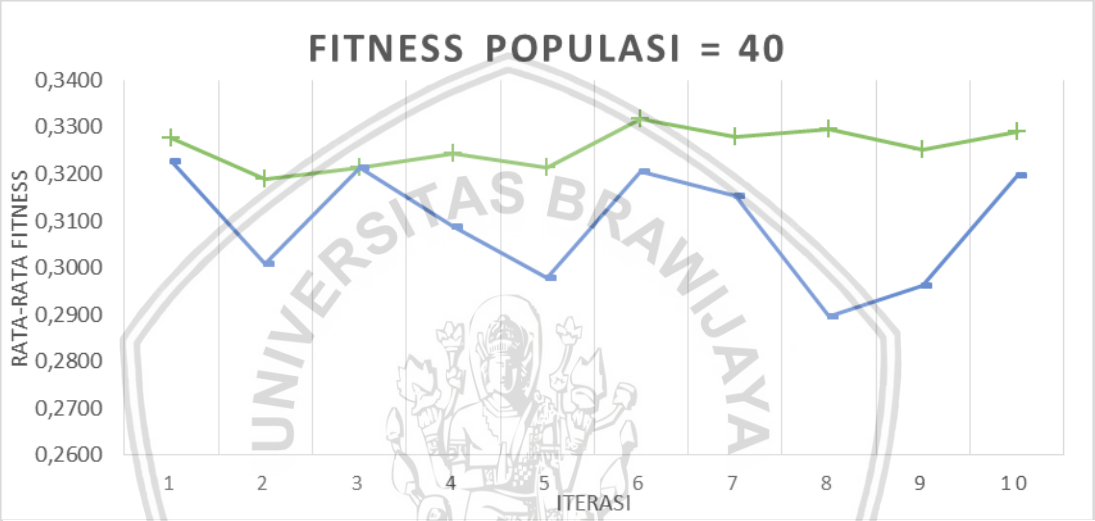
no	Iterasi = 1	Iterasi = 2	Iterasi = 3	Iterasi = 4	Iterasi = 5	Iterasi = 6	Iterasi = 7	Iterasi = 8	Iterasi = 9	Iterasi = 10
1	0,3136	0,3189	0,2869	0,3327	0,3213	0,3156	0,3433	0,3544	0,2769	0,3050
2	0,3432	0,3525	0,3209	0,3168	0,3037	0,3263	0,3245	0,3419	0,3160	0,3229
3	0,3128	0,3233	0,3177	0,3328	0,3213	0,3570	0,3364	0,3379	0,1115	0,3241
4	0,3356	0,3347	0,3439	0,3297	0,2599	0,3169	0,3533	0,1522	0,3215	0,3241
5	0,3243	0,2489	0,3224	0,3296	0,3425	0,3267	0,3346	0,3426	0,3153	0,3281
6	0,3176	0,3188	0,3388	0,3261	0,3241	0,3372	0,3596	0,3269	0,3207	0,2500
7	0,3396	0,3144	0,3245	0,3508	0,1841	0,3645	0,2930	0,3376	0,3020	0,3392
8	0,3378	0,1696	0,3204	0,3410	0,2997	0,3492	0,3521	0,3273	0,3293	0,3369
9	0,3276	0,3132	0,3066	0,1195	0,3276	0,3387	0,1264	0,1143	0,3232	0,3352
10	0,2755	0,3144	0,3315	0,3083	0,2951	0,1736	0,3294	0,2619	0,3447	0,3301

Tabel 6.14 Rata-Rata *Fitness* GA dan SA untuk *Popsize* = 40

Iterasi	Rata-Rata *Fitness* GA	Rata-Rata *Fitness* GA-SA
1	0,3277	0,3228
2	0,3189	0,3009
3	0,3214	0,3213
4	0,3244	0,3087

Tabel 6.15 Rata-Rata *Fitness* GA dan SA untuk *Popsize* = 40 (lanjutan)

5	0,3212	0,2979
Iterasi	Rata-Rata *Fitness* GA	Rata-Rata *Fitness* GA-SA
6	0,3316	0,3206
7	0,3279	0,3153
8	0,3294	0,2897
9	0,3251	0,2961
10	0,3289	0,3196



Gambar 6.4 Grafik Rata-Rata *Fitness* GA dan GA-SA Untuk *Popsize* = 40

Keterangan :

Garis Biru = Rata – Rata *Fitness* Algoritme Genetika dengan *Simulated annealing*

Garis Hijau = Rata – Rata *Fitness* Algoritme Genetika

Dari grafik yang ditunjukkan oleh gambar 6.4 bisa diambil kesimpulan bahwa untuk jumlah populasi = 30 Algoritme Genetika dengan *Simulated annealing* selalu menghasilkan optimasi yang lebih buruk daripada algoritme genetika saja, selain itu keduanya memiliki *Fitness* yang cukup tinggi yang berarti dengan *Popsize* 40 keduanya menghasilkan kombinasi pakan dengan harga yang murah dan nutrisi yang baik dalam artian nutrisi melebihi kebutuhan sapi.

6.2.5 Pengujian Dengan *Popsize* 50

Pengujian yang pertama *Popsize* di set dengan nilai 50 dan iterasi di set dengan nilai 1-10 dimana setiap jumlah iterasi akan dilakukan pengujian 10 kali kemudian diambil nilai rata-rata nya, misal untuk iterasi = 1 dilakukan pengujian sebanyak 10 kali kemudian diambil rata-rata nilai *Fitness*nya, setelah itu dilakukan pengujian untuk iterasi = 2 dilakukan

pengujian sebanyak 10 kali kemudian diambil rata-rata nilai *Fitness*nya dan seterusnya hingga iterasi = 10. Hasil Pengujian ditunjukkan oleh tabel 6.13, tabel 6.14, tabel 6.15 dan gambar 6.5.

Tabel 6.16 Hasil Pengujian Algoritme Genetika Untuk *Popsize* = 40

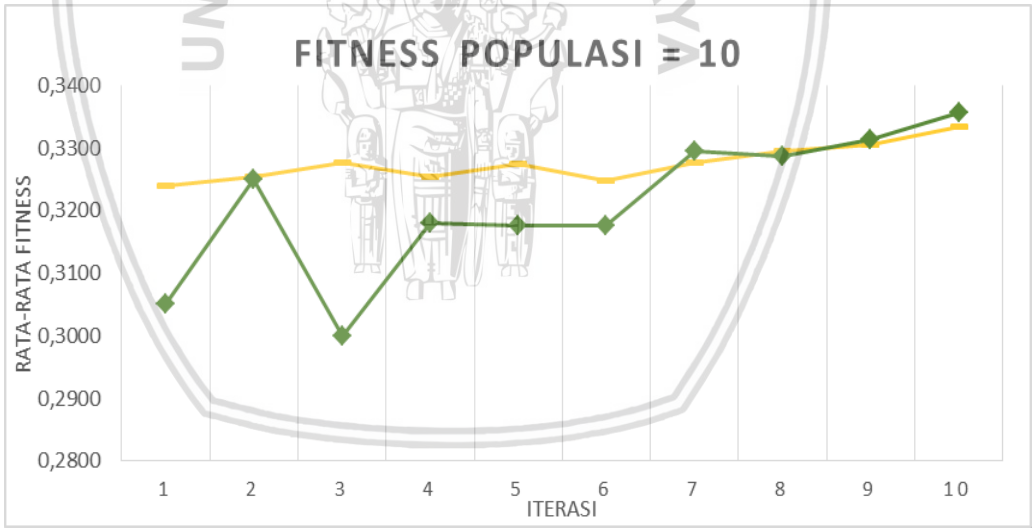
no	Iterasi = 1	Iterasi = 2	Iterasi = 3	Iterasi = 4	Iterasi = 5	Iterasi = 6	Iterasi = 7	Iterasi = 8	Iterasi = 9	Iterasi = 10
1	0,3160	0,3180	0,3387	0,3013	0,3280	0,3414	0,3123	0,3295	0,3348	0,3296
2	0,3358	0,3438	0,3220	0,3067	0,3155	0,3199	0,3303	0,3304	0,3295	0,3223
3	0,3282	0,3256	0,3095	0,3281	0,3245	0,3238	0,3299	0,3471	0,3476	0,3376
4	0,3165	0,3267	0,3301	0,3225	0,3217	0,3136	0,3313	0,3295	0,3285	0,3334
5	0,3259	0,3261	0,3108	0,3484	0,3249	0,3314	0,3228	0,3221	0,3183	0,3370
6	0,3302	0,3141	0,3264	0,3191	0,3137	0,3318	0,3111	0,3384	0,3177	0,3600
7	0,3258	0,3284	0,3481	0,3157	0,3483	0,3392	0,3216	0,3249	0,3292	0,3328
8	0,3275	0,3233	0,3358	0,3283	0,3356	0,3110	0,3363	0,3101	0,3426	0,3166
9	0,3086	0,3245	0,3112	0,3374	0,3313	0,3290	0,3472	0,3312	0,3218	0,3355
10	0,3256	0,3236	0,3446	0,3471	0,3303	0,3064	0,3343	0,3307	0,3358	0,3294

Tabel 6.17 Hasil Pengujian *Simulated annealing* Untuk *Popsize* = 40

no	Iterasi = 1	Iterasi = 2	Iterasi = 3	Iterasi = 4	Iterasi = 5	Iterasi = 6	Iterasi = 7	Iterasi = 8	Iterasi = 9	Iterasi = 10
1	0,3206	0,2421	0,3341	0,3223	0,3286	0,3466	0,3319	0,3322	0,3380	0,2527
2	0,3378	0,3466	0,3238	0,3082	0,3448	0,3278	0,3539	0,3369	0,3486	0,3230
3	0,1447	0,3311	0,3116	0,3303	0,3379	0,3255	0,3326	0,3471	0,3493	0,3478
4	0,3439	0,3280	0,1539	0,3233	0,3269	0,3250	0,3384	0,3281	0,3334	0,3358
5	0,3317	0,3348	0,3246	0,3509	0,3331	0,3318	0,3328	0,3230	0,3055	0,3431
6	0,3352	0,3196	0,3409	0,3262	0,3417	0,3330	0,3160	0,3404	0,3194	0,3340
7	0,2511	0,3299	0,2726	0,3239	0,3522	0,3472	0,3234	0,3257	0,3392	0,2668
8	0,3407	0,3547	0,3362	0,3298	0,1451	0,3113	0,3367	0,3140	0,3586	0,3322
9	0,3117	0,3270	0,2483	0,3448	0,3322	0,3298	0,3539	0,3343	0,3264	0,3378
10	0,3332	0,3366	0,3534	0,2209	0,3341	0,3169	0,2665	0,3322	0,3384	0,3334

Tabel 6.18 Rata-Rata *Fitness* GA dan SA untuk *Popsize* = 40

Iterasi	Rata-Rata *Fitness* GA	Rata-Rata *Fitness* GA-SA
1	0,3240	0,3248
2	0,3254	0,3277
3	0,3277	0,3294
4	0,3255	0,3306
5	0,3274	0,3334
Iterasi	Rata-Rata *Fitness* GA	Rata-Rata *Fitness* GA-SA
6	0,3051	0,3177
7	0,3251	0,3295
8	0,2999	0,3286
9	0,3180	0,3314
10	0,3177	0,3357



Gambar 6.5 Grafik Rata-Rata *Fitness* GA dan GA-SA Untuk *Popsize* = 50

Keterangan :

Garis Hijau = Rata – Rata *Fitness* Algoritme Genetika dengan *Simulated annealing*

Garis Kuning = Rata – Rata *Fitness* Algoritme Genetika

Dari grafik yang ditunjukkan oleh gambar 6.5 bisa diambil kesimpulan bahwa untuk jumlah populasi = 30 Algoritme Genetika dengan *Simulated annealing*sebagian besar menghasilkan optimasi yang lebih buruk daripada algoritme genetika saja, selain itu keduanya memiliki *Fitness* yang cukup tinggi yang berarti dengan *Popsize* 50 keduanya menghasilkan kombinasi.

BAB VII PENUTUP

7.1 Kesimpulan

Kesimpulan yang diperoleh melalui hasil uji coba yang telah dilakukan mengenai penerapan algoritme genetika - *simulated annealing* untuk menyelesaikan permasalahan optimasi komposisi pakan untuk penggemukan sapi yaitu sebagai berikut:

1. Algoritme genetika - *simulated annealing* dapat diterapkan pada permasalah optimasi pakan untuk penggemukkan sapi potong dengan menggunakan representasi kromosom secara *real code*, metode *extended intermediate*, menggunakan *random mutation* serta penyeleksian dengan metode *elitism selection*.
2. Algoritme genetika sebenarnya sudah menghasilkan output yang baik yaitu komposisi pakan dengan nutrisi yang baik dan harga yang rendah, tetapi ternyata dengan ditambahkan *simulated annealing* dapat menghasilkan hasil yang lebih baik.
3. Dari pengujian yang telah dilakukan ternyata semakin kecil nilai *Popsize* dan iterasi, semakin besar nilai tn dan t0 maka semakin besar kemungkinan algoritme genetika - *simulated annealing* untuk mendapatkan hasil yang lebih baik daripada algoritme genetika saja, sebaliknya semakin besar nilai *Popsize* dan iterasi, semakin kecil nilai tn dan t0 maka semakin kecil pula kemungkinan algoritme genetika - *simulated annealing* untuk mendapatkan hasil yang lebih baik dari algoritme genetika saja.

7.2 Saran

Pada penelitian ini, terdapat hal yang dapat ditambahkan dan dikembangkan untuk penelitian selanjutnya antara lain :

1. Pada program ini tidak dapat menambahkan data baru berupa data sapi, data pakan dan data nutrisi. Oleh karena itu, lebih baik dalam program ini dapat menambahkan fitur tersebut untuk mempermudah si peneliti maupun si peternak apabila ingin menambahkan data yang baru ke dalam database.
2. Pada penelitian ini hanya menggunakan kombinasi pakan maksimal 4 macam jenis pakan yaitu, 1 untuk hijauan dan 3 untuk konsentrat. Sebaiknya jumlah kombinasi tersebut dapat ditambahkan agar dapat memperoleh kombinasi yang beragam dengan harapan mendapatkan harga yang lebih murah.
3. Pada program ini sebaiknya dapat melakukan pemillihan jenis pakan secara otomatis dalam menggemukkan sapi dengan penambahan berat badan tertentu agar dapat menghasilkan solusi lain yang optimal.
4. Jumlah bahan pakan bisa ditambahkan karena dalam permasalahan ini hanya terdapat 25 jenis bahan pakan.

DAFTAR PUSTAKA

Anggarsari, Fitri. 2017. *Optimasi Kebutuhan Gizi untuk Balita Menggunakan Hybrid Algoritme genetika - simulated annealing.* Fakultas Ilmu Komputer. Universitas Brawijaya. Malang.

Aryani, Amalia Kartika. 2017. *Hibridisasi Algoritme genetika - simulated annealing untuk Optimasi Multi-Trip Vehicle Routing Problem with Time Windows (Studi Kasus: Pariwisata Kabupaten Banyuwangi).* Fakultas Ilmu Komputer. Universitas Brawijaya. Malang.

Ardina, Shelly Puspa. 2017. *Optimasi Jumlah Pinjaman Koperasi Menggunakan Fuzzy Tsukamoto Dengan Algoritme Genetika.* Fakultas Ilmu Komputer. Universitas Brawijaya. Malang.

Efendi, Mahbub Zaeni, Retno Novi Dayawati, Agung Toto Wibowo. 2009. *SISTEM PENJADWALAN KULIAH ITTELKOM DENGAN HYBRID ALGORITME GENETIKA SIMULATED ANNEALING (GA-SA).* Fakultas Teknik Informatika. Universitas Telkom.

Fakhiroh, Dorrotul. 2017. *Optimasi Komposisi Pakan Sapi Perah Menggunakan Algoritme Genetika.* Fakultas Ilmu Komputer. Universitas Brawijaya. Malang

Laryska, Nabila, Tri Nurhajati. 2013. *PENINGKATAN KADAR LEMAK SUSU SAPI PERAH DENGAN PEMBERIAN PAKAN KONSENTRAT KOMERSIAL DIBANDINGKAN DENGAN AMPAS TAHU.* Fakultas Kedokteran Hewan. Universitas Airlangga. Surabaya.

Otoluwa, Moh. Andri. 2016. *PROSPEK PENGEMBANGAN USAHA TERNAK SAPI POTONG DI KECAMATAN BOLANGITANG TIMURKABUPATEN BOLAANG MONGONDOW UTARA.* Fakultas Peternakan. Universitas Sam Ratulangi. Manado

Suryana. 2009. *PENGEMBANGAN USAHA TERNAK SAPI POTONG BERORIENTASI AGRIBISNIS DENGAN POLA KEMITRAAN.* Banjarbaru.

Wahyono, Teguh, Suharyono, Irawan Sugoro. 2011. *INOVASI PAKAN KOMPLIT TERHADAP PERTAMBAHAN BERAT BADAN HARIAN TERNAK SAPI PERANAKAN ONGOLE JANTAN.*

Wiyatna, M. Fatah, E.Gunardi, K. Mudikdjo. 2012. *Produktivitas Sapi Peranakan Ongole pada Peternakan Rakyat di Kabupaten Sumedang (Productivity of Peranakan Ongole Cattle on traditional farm system in Sumedang Region).* Fakultas Peternakan. Institut Pertanian Bogor.